VASIC: VALIDATION-GUIDED ADAPTATION FOR SYMBOLIC INTEGRATION WITH LLMS AND CAS

Anonymous authors
Paper under double-blind review

ABSTRACT

AI systems are increasingly used in research and education, generating complex proofs, derivations, and results faster than experts can reliably validate. Interactive theorem provers such as Lean provide strong guarantees for formalized proofs, but fully formalizing the symbolic derivations and analytical calculations central to physics and engineering is often impractical. As a result, a growing volume of LLM-generated mathematics in these domains remains unverified. To bridge this gap, we propose a complementary, scalable method based on Computer Algebra Systems (CAS) to improve the reliability of LLM-generated mathematical outputs. As a proof of concept, we present VASIC, a validation-guided LLM-CAS protocol for symbolic integration. VASIC tests each candidate LLM solution using symbolic and numerical methods. Failed attempts are routed through feedback-guided adaptation and retried. We benchmark VASIC against standalone CAS and LLM baselines from a broad range of model families and versions. Compared with direct use of the same model, VASIC increases validated success rates by 3.5–10 percentage points for frontier models and by up to 36.2 percentage points for smaller and less expensive models. For example, DeepSeek Reasoner V3 achieves 84.0% validated success with full VASIC, close to DeepSeek V4's 85.2%, at roughly half the estimated model-inference cost, counting the additional LLM requests made by VASIC. Together, these results show that LLM-CAS hybrids can increase AI success rate and practical reliability in problems of general importance for science and engineering.

1 INTRODUCTION

Integrals are fundamental to science and engineering, supporting both theoretical research and practical applications. Although numerical integration can approximate many quantities, closed-form symbolic integrals remain critical for simplification, downstream derivations, parameterized analysis, and exact reasoning (Rich et al., 2018).

Symbolic integration - the task of finding exact antiderivatives of indefinite integrals or evaluating definite integrals in symbolic form - has long been a central problem in computer algebra (von zur Gathen & Gerhard, 2013). Building on foundational work in differential algebra, Risch formulated an algorithmic approach to construct elementary antiderivatives (Risch, 1969). Subsequent work extended these ideas to algebraic, transcendental, and special-function expressions, laying the theoretical and practical groundwork for modern CAS integral solvers (Geddes et al., 1992). General-purpose CAS such as *Mathematica* and *Maple* combine exact transformations, heuristics, and differential-algebraic methods to solve broad classes of integrals (Wolfram Research, Inc., 2025; Monagan et al., 2005; von zur Gathen & Gerhard, 2013). Specialized tools instead focus on domain-specific structure. For Feynman integrals, parametric representations and hyperlogarithmic forms enable the solution of narrower but physically important integral families (Panzer, 2015).

Even so, no single procedure is complete across the expressions encountered in practice. The Risch algorithm is complete for elementary integration under suitable algebraic assumptions (Risch, 1969; Moses & Risch, 1980), but it is not applicable in many important practical scenarios. In practice, most systems use a combination of heuristic algorithms and transformations. The success of existing solvers thus depends on expression representation (Bronstein, 2005).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

This had been the state-of-the-art in integral solving until recently. With the arrival of LLMs, the space of integrals that can be solved by computers expanded once more. LLMs can generate candidate antiderivatives and alternative representations, including approaches not attempted by any particular CAS or specialized solver. Indeed, certain LLMs have been shown to solve variants of symbolic-mathematics problems that the established CAS, such as SymPy and Mathematica, failed to solve (Shalyt et al., 2025). Despite their potential, LLMs make mistakes and cannot provide certainty about the correctness of their candidate solutions or transformations (Kalai et al., 2025). There is a need for an approach that harnesses the advantages of LLMs with the computational rigor of established CAS.

Here we introduce VASIC (**V**alidation-Guided **A**daptation for **S**ymbolic **I**ntegration with LLMs and **C**AS), a validation-guided LLM-CAS protocol for symbolic integration. We develop a CAS-based validator that checks LLM-generated solutions and verifies that candidate transformations are equivalent to the original integral. If the check does not pass, its outcome directs the LLM to try again or take a different approach. The validation protocol combines symbolic checks with complementary numerical evidence to determine whether candidate solutions are valid. Figure 1 summarizes the general VASIC architecture. Our experiments and benchmark tests confirm that errors occur even among strong models, supporting the case for external validation. An interactive demonstration of VASIC is available online.[1]

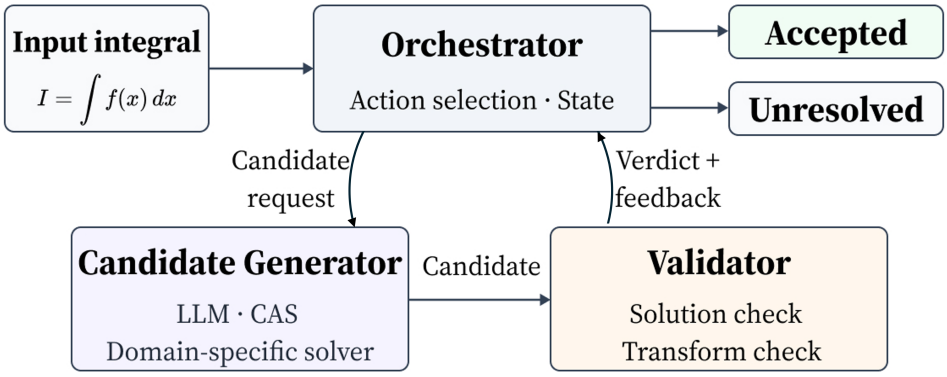


Figure 1: **VASIC architecture.** The orchestrator selects an action and a candidate generator, then applies the predefined acceptance policy to the resulting output.

We evaluate VASIC against 12 LLMs using the same integration problems and report validated success rates and estimated model-inference costs. Compared with direct use of the same model, full VASIC increases validated success rate by 3.5–10 percentage points for the latest tested frontier models and by up to 36.2 percentage points for smaller or earlier-generation models.

Surprisingly, once used in our hybrid solver, smaller, open-weight, and lower-cost models achieve an overall success rate exceeding that of frontier LLMs without VASIC and approaching the performance of frontier models used with VASIC.

Looking at the bigger picture, symbolic integration is a hallmark example of a broader idea for hybrid trust models that can enhance AI for math. A similar approach can apply whenever finding a solution is difficult but obtaining independent evidence of its correctness is comparatively straightforward, whether through automated formal verification, CAS-based validation, or approximate or probabilistic checks. In such cases, the general VASIC approach exploits this asymmetry and provides a kind of "orthogonal" soft validation, using evidence independent of the generating model. This approach is especially useful for the many cases in mathematics, physics, and engineering in which formal verification is not yet sufficient, with a vision of longer-term hybrids of LLMs, formal verification, and softer CAS-based validation.

[1] `https://zweb-blue.vercel.app/`

2 Related Work

Attempts to use machine learning for symbolic integration began before the rise of LLMs. Lample & Charton (2020) showed that sequence-to-sequence models trained on synthetic expressions can solve symbolic mathematics tasks, including integration and solving differential equations. Noorbakhsh et al. (2022) studied whether pretraining on natural-language machine translation helps models learn symbolic-mathematics tasks, including symbolic integration and differential-equation solving. These approaches treat symbolic integration as end-to-end prediction, and train these models on synthetically generated integrand-antiderivative pairs rather than on examples drawn from real-world mathematical practice. Other work, such as SIRD and AlphaIntegrator, uses learned guidance within symbolic systems to select among integration steps and routines (Sharma et al., 2023; Unsal et al., 2024).

Taken together, these learned-integration approaches train a model either to predict an antiderivative directly, select the next symbolic rule, or choose among integration algorithms available inside a CAS. Their scope and flexibility are limited by the training data, model architecture, and mathematical representations built into each system.

Recent general-purpose reasoning models, without integration-specific training, have achieved strong performance on symbolic-integration benchmarks. Several recent benchmarks evaluate such general-purpose LLMs: IntegralBench evaluates LLMs on definite-integral problems (Tang et al., 2025). ASyMOB covers symbolic operations including integration, limits, differential equations, and series, and evaluates robustness to input perturbations (Shalyt et al., 2025). Other benchmarks, such as U-MATH and UGMathBench, evaluate LLMs on a broader range of problems, including integration, linear algebra, and differential equations (Chernyshev et al., 2025; Xu et al., 2025).

Across these benchmarks, model outputs are evaluated relative to provided reference answers. This introduces potential evaluation errors, as reference annotations may contain incorrect answers or formatting errors, while valid answers may take mathematically equivalent forms that differ from the reference (Xu et al., 2025; Fang et al., 2025). LLM-assisted grading introduces a related limitation. μ-MATH shows that solution judgment on mathematical solutions remains challenging even for strong models (Chernyshev et al., 2025).

Therefore, neither matching a reference answer nor an LLM's own evaluation is enough to establish mathematical correctness. Motivated by these limitations, VASIC uses external validation to check a candidate directly against the original problem rather than relying on reference matching or model judgment. Such validation can accept solutions that differ in form but are mathematically equivalent, without requiring a reference answer.

Self-Refine (Madaan et al., 2023) shows that iterative refinement using model-generated feedback can improve outputs at test time. However, Huang et al. (2024) show that, without external feedback or verification, LLMs often fail to improve their reasoning and may revise initially correct answers into incorrect ones. These findings motivate the use of validators, tools, or other external sources of feedback. Related work on process supervision shows that evaluating intermediate reasoning steps can outperform supervision based only on final answers for mathematical reasoning (Lightman et al., 2024).

VASIC builds on these lines of work by applying external feedback and intermediate validation to symbolic integration. Related tool-assisted approaches use external tools in different ways: CRITIC uses tool-interactive critiques, while CSV and SymCode use model-generated executable code in their verification and correction loops (Gou et al., 2024; Zhou et al., 2024; Bagheri Nezhad et al., 2026). VASIC instead uses a CAS-based validator, defined independently of the generating model, to assess both candidate solutions and proposed transformations; the model cannot accept its own revised answers. Unlike AlphaIntegrator, VASIC requires no integration-specific training or learned search over a symbolic action space (Unsal et al., 2024). Thus, the LLM proposes and revises antiderivatives and transformations, while the validation protocol governs acceptance.

3 THE VASIC FRAMEWORK

At the center of VASIC is a validation-guided orchestration loop. At each step, the orchestrator selects an action and a solver, receives the resulting output, and routes it through the appropriate acceptance or validation procedure (Figure 1). The complete general procedure is given in Appendix A, Algorithm 1. Figure 2 shows examples of validator-guided retry (a) and transformation validation (b). The framework leaves the orchestration policy open. Here we evaluate the deterministic route in Algorithm 2 and study how validated feedback and representation changes affect performance.

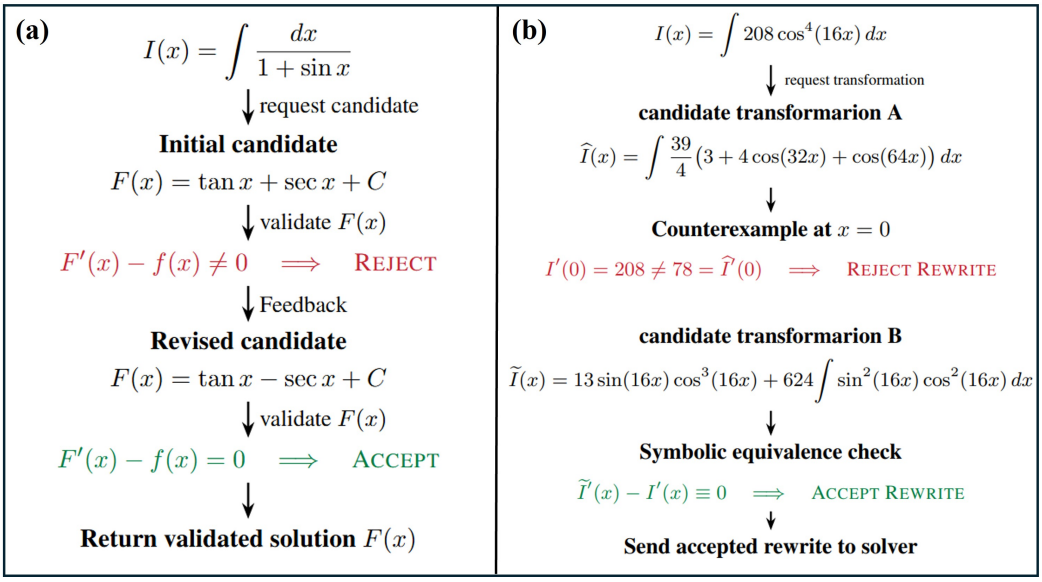


Figure 2: **VASIC examples. (a) Validator-guided retry.** A rejected candidate is revised using validator feedback. The revised candidate is re-validated and accepted. **(b) Transformation validation.** The panel contrasts two transformation candidates for the same integral. VASIC rejects a non-equivalent candidate and passes an equivalent one to the downstream solver.

Our evaluated implementation uses SymPy and Wolfram as CAS backends for answer and transformation validation. All outputs are routed through VASIC's assessment procedure, whose acceptance policy depends on the candidate source. Explicit results returned by the CAS solvers are accepted directly because, in this implementation, the CAS solvers are treated as trusted symbolic-computation components. By contrast, LLM-generated candidates are treated as untrusted and must pass external CAS-based validation.

VASIC uses pretrained general-purpose LLMs as candidate generators within a modular LLM-CAS framework. This allows newer models, and models that were trained on a broader range of data and mathematical problems, to be incorporated as they become available.

The LLM route begins with a single attempt to solve the original integral, which we call the direct solve. Solutions validated at this route are counted as direct validated solves. Problems that remain unsolved or unvalidated receive up to two adaptation attempts using structured validator feedback; solutions first validated during these attempts are counted as feedback saves. If these attempts fail, VASIC tries different transformation types, each with a separate budget of up to three attempts. Candidate transformations must pass equivalence validation before downstream solving, and solutions first validated through these routes are counted as transformation saves. VASIC solves each accepted transformation using the same procedure as the original integral: one attempt and up to two feedback retries in a shared context window within 20 minutes. It validates each answer against the original integral. The evaluated route uses fixed attempt budgets. For problems that reach the transformation route, all five transformation types are evaluated to support comparison among them; a deployment-oriented policy could stop after the first accepted solution. The complete budgets,

stopping rules, and pseudocode are provided in Appendix A, Algorithm 2. Concrete feedback-guided and transformation-guided examples are presented in Appendix D.

As described above, LLM-generated candidates must pass external validation. For an indefinite integral with integrand $f(x)$ and candidate $F(x)$, the validator checks whether $F'(x)-f(x)$ simplifies to zero. It returns PASS when symbolic checking establishes equality unless numerical checking contradicts it; it does not accept numerical agreement alone. The validator returns FAIL when it detects a mismatch in the derivative or definite-integral value.

For a definite integral, the validator compares the candidate with the exact value computed by the CAS. If the CAS cannot obtain an exact value, the validator uses numerical comparison and may return PASS from numerical agreement alone. A backend returns INCONCLUSIVE when it lacks sufficient evidence or its symbolic and numerical checks conflict. For answer validation, VASIC adopts the first conclusive backend verdict, PASS or FAIL, and consults another backend only when the first returns INCONCLUSIVE. Here, independent means separate from the generating LLM, not confirmation by both CAS backends.

An item enters the transformation route only after the direct attempt and all validator-guided adaptation attempts end without a PASS. Rather than requesting another answer, this route asks the model to produce an equivalent representation of the original integral. The evaluated implementation uses exactly five transformation types: integration by parts, integrand simplification, partial-fraction decomposition, trigonometric rewriting, and substitution. The general framework permits other transformation menus, but we evaluate only these five types in this study. Each transformation type allows at most three model attempts, including attempts that use validator feedback. VASIC checks each transformation's structure and mathematical equivalence before solving it. In the reported experiments, a conclusive CAS pass sufficed for acceptance, including a numerical whole-integral equivalence pass for definite integrals. Appendix B.5 provides further details.

Figure 3 shows numerical checks in two recorded validator-guided retries: one for a definite-integral candidate and one for a transformation before downstream solving.

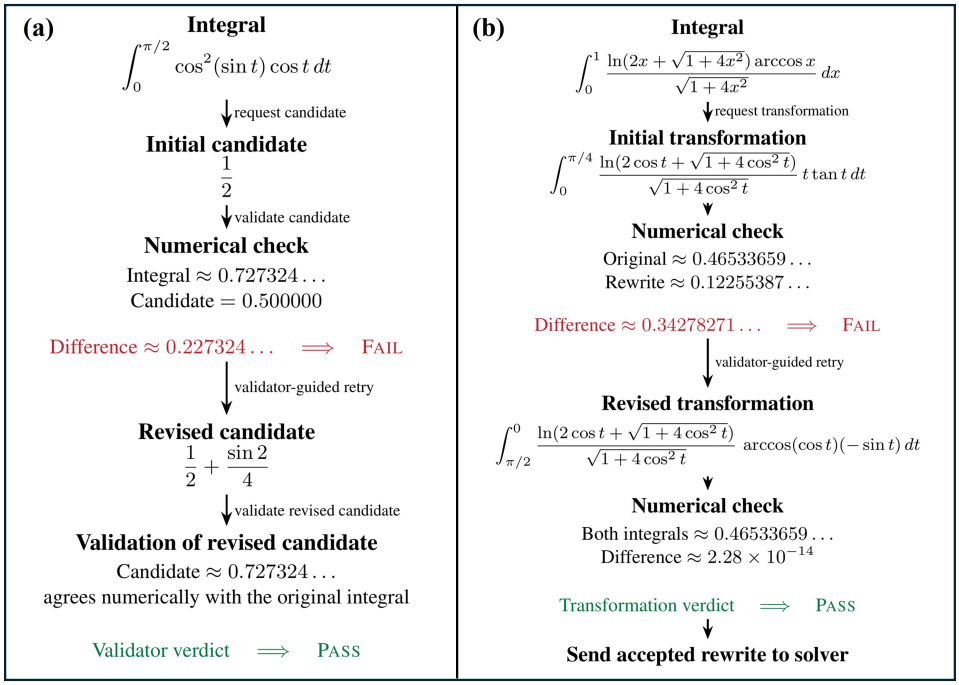


Figure 3: **Numerical checks for definite integrals.** (a) The initial candidate differs numerically from the integral; the revised candidate agrees and passes validation. (b) The initial transformation changes the integral's numerical value; the revised transformation preserves it, passes the whole-integral equivalence check, and is sent to the downstream solver.

4 RESULTS

4.1 BENCHMARK AND EVALUATION

We evaluate VASIC on a set of $N = 1{,}445$ symbolic integration tasks, comprising both indefinite antiderivatives and definite integrals. Each task includes any assumptions or bounds required by the validation protocol. The evaluation set combines benchmark-derived problems from IntegralBench, the integral-calculus track of U-MATH, ASyMOB, CUMath, MathOdyssey, and UGMathBench (Tang et al., 2025; Chernyshev et al., 2025; Shalyt et al., 2025; Tran et al., 2025; Fang et al., 2025; Xu et al., 2025) with expert and classical problems drawn from Vălean's integral collections and the MIT Integration Bee (Vălean, 2019; 2023; MIT Mathematics, 2026). These sources were filtered and deduplicated before evaluation. Table 3 in Appendix C.2 reports the composition of the evaluation set. No model was trained or fine-tuned on the evaluation set, although prior exposure through pretraining cannot be ruled out (Section 5).

Prompts and request configuration. Each LLM request was limited to 480 seconds, while each solve job had a total wall-time limit of 1,200 seconds. The direct route uses a fixed base solver prompt. Each of the two feedback-guided adaptation retries uses the same prompt augmented with structured feedback from the prior non-pass.

Metrics and statistical analysis. For each method, we report the validated-solution rate, defined as the proportion of eligible problems receiving a primary-metric PASS, and the additional problems solved through each route. All other outcomes, including FAIL, INCONCLUSIVE, timeout, parsing error, and infrastructure error, are counted as non-successes. A non-success does not imply that the problem has no valid mathematical solution. Validated success includes symbolic passes and, for definite integrals, numerical passes when exact evaluation is unavailable.

4.2 VASIC SUCCESS RATES

We ran Wolfram and SymPy on all benchmark integrals, and at least one of them returned an explicit result for 835 (57.8%). CAS outputs were scored under the CAS evaluated-return rule, not the validation policy used for LLM answers, so we report this rate only as context.

Figure 4 reports direct validated success, the additional gains from the feedback and transformation routes, and final VASIC success. Feedback saves are problems whose first solution to pass validation is produced during validator-guided adaptation, while transformation saves are problems whose first passing solution is produced after an equivalence-validated transformation. The final VASIC success rate is the proportion of problems solved through direct solving, feedback-guided adaptation, or validated transformations. The relative contributions vary substantially across systems: feedback accounts for most of the gain for the three strongest direct solvers, while transformations account for a larger share for several weaker models. DeepSeek Reasoner V3 shows the largest increase, from 47.8% to 84.0% (+36.2 pp). Detailed validated success rates for all twelve LLM systems are reported in Table 1 in Appendix C.

4.3 MODEL-INFERENCE COSTS

VASIC solves more problems, but doing so requires additional model calls. We compare this added cost across models. Figure 4(b) compares final VASIC success with the number of additional validated problems per USD of additional estimated model-inference cost. DeepSeek Reasoner V3 combines high final VASIC success with relatively high cost efficiency. Mistral Large and Llama 4 Maverick solve the most additional problems per USD, whereas the frontier models reach higher final success with fewer additional problems per USD. Table 7 in Appendix C.4 lists the estimated costs for direct solving and full VASIC, together with the additional cost over direct solving. These estimates exclude CAS execution and do not measure end-to-end latency.

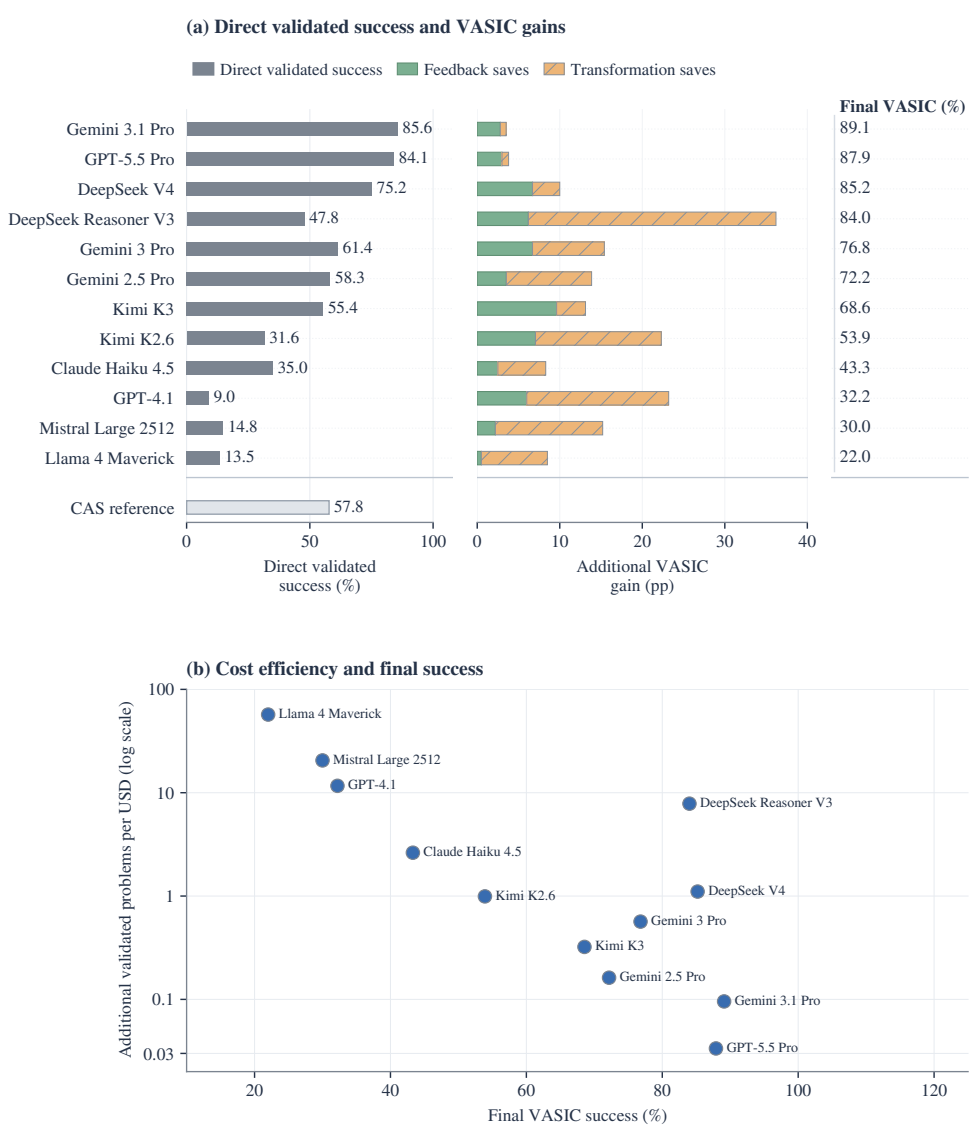


Figure 4: **Validated success rates and inference-cost efficiency of VASIC. (a)** Direct success (left), additional gains from the feedback and transformation routes (center), and final VASIC success (right). Gains are reported in percentage points (pp), and final success is direct success plus both gains. **(b)** Final VASIC success versus additional validated problems per USD of additional estimated model-inference cost, relative to direct solving. The vertical axis uses a logarithmic scale. Costs cover every model request, including failed attempts, and exclude CAS execution. The CAS row is an operational reference: 835 problems (57.8%) produced explicit results, accepted without external validation under a different acceptance policy. Direct CAS returned explicit results for 6 problems that Gemini 3.1 Pro did not validate through full VASIC, 29 for GPT-5.5 Pro, and 16 for DeepSeek V4. For direct LLM solving alone, the corresponding counts were 18, 45, and 36.

4.4 Effect of validator-guided feedback under matched retries

VASIC increases validated success partly by prompting LLMs to try again after an unsuccessful direct attempt. However, repeated attempts can also solve additional problems without feedback. To determine whether VASIC provides value beyond such naive retries, we compare feedback-guided retries with matched retries that receive no validator feedback. Figure 5 shows that validator feedback

improves validated success for all three evaluated models under the same maximum call budget and retry time limits.

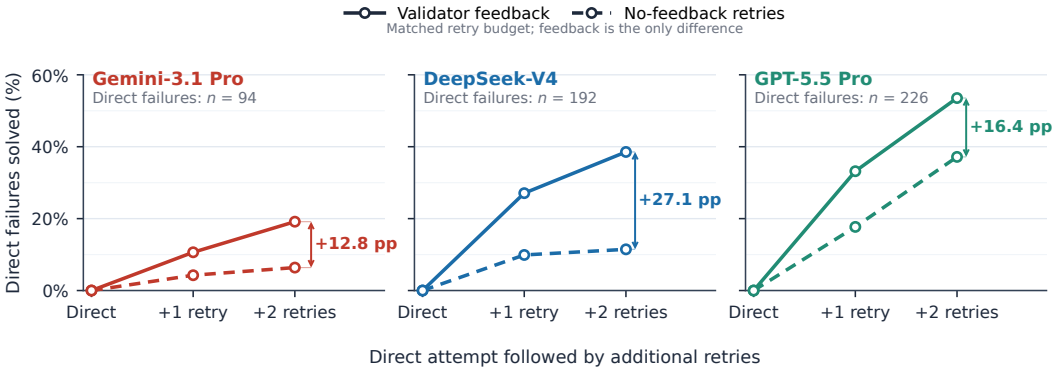


Figure 5: **Benefit of validator feedback beyond simple retries.** Both branches are evaluated on the same problems that failed the initial direct attempt. Each branch allows up to two retries, each in a separate context window with a 20-minute limit. The y-axis shows the percentage of eligible problems solved and validated by the end of each retry. Solid lines show feedback-guided retries, and dashed lines show retries without feedback. After two retries, the feedback-guided and no-feedback branches validated 18 and 6 problems for Gemini 3.1 Pro ($n = 94$), 74 and 22 for DeepSeek V4 ($n = 192$), and 121 and 84 for GPT-5.5 Pro ($n = 226$). Arrows show the final difference in percentage points.

Among the 512 problems that failed the initial direct attempt and were evaluated under both conditions, feedback-guided retries yielded validated solutions for 213 problems (41.6%) after up to two retries, compared with 112 problems (21.9%) for matched retries without feedback. Both conditions allowed at most two retries with the same retry time limits, although realized token consumption could differ. This provides a controlled comparison of validator-guided feedback against retry-only sampling. Detailed final outcomes, including problems solved exclusively by either branch, are reported in Table 2 in Appendix C.

4.5 Validator reliability

Because VASIC accepts solutions based on validator decisions, we evaluate the validator separately from the end-to-end system. We drew 100 cases from each of four pools: correct answers, incorrect answers, equivalent transformations, and non-equivalent transformations. The incorrect answers were produced by sign flips, factor-of-two changes, and dropped terms. No incorrect answer or non-equivalent transformation received PASS. The validator returned INCONCLUSIVE on 16 of the 400 cases.

As a second check, we drew a stratified sample of 1,037 accepted records and rechecked each one with the CAS that had not accepted it. The second CAS confirmed 794 records, contradicted 3, and was inconclusive on 240. Manual review of the three contradictions found one incorrect acceptance; the other two records were mathematically valid. Because the sample was balanced across acceptance modes and originating CAS, these pooled counts do not estimate ledger-wide error rates. Appendix C gives the complete breakdown, the sampling procedure, and additional failure analysis.

5 Limitations

VASIC uses exact symbolic checks when possible and numerical checks for definite integrals when the CAS cannot obtain an exact value. Numerical checks provide a weaker but still useful form of validation. The system remains limited by the capabilities of CAS. Some correct candidates remain unvalidated when the available CAS cannot establish the required identity. For example, highly oscillatory definite integrals can be difficult to verify numerically. Integrals over the complex plane still require advanced analytical methods to handle branches and other classical complications. All of these imply that ongoing developments in symbolic and numerical solvers could be used to enhance

VASIC. The reported success rate therefore reflects both the model's ability to find a solution and the validator's ability to verify it.

Runtime is another limitation, as the cost of LLM and CAS calls can be difficult to predict before execution, making fixed timeouts either stop valid computations or waste resources. Adaptive compute allocation in LLMs and algorithm selection in CAS offer promising ways to manage this variability (Snell et al., 2025; Barket et al., 2024). Such methods could help VASIC allocate computation more effectively and avoid unnecessary timeouts.

6 Discussion and Outlook

Our experiments reveal a clear difference in where VASIC's gains come from across models. Feedback accounts for most of the improvement among the strongest models, whereas representation transformations account for a larger share for several weaker models. This pattern may reflect a broader shift in the kinds of difficulties LLMs face as their capabilities improve. The models also differ in their capabilities, costs, and the efficiency with which additional computation produces validated solutions. These differences show that model choice and orchestration strategy should be considered together.

Rather than train a model specifically for integration, as most earlier AI approaches did, VASIC uses existing LLMs to generate candidates and combines them with established CAS and CAS-based validation tools that we developed. This approach makes VASIC modular and independent of the particular LLMs used, leaving the same CAS-based validation in place and relevant even as new AI models become available. In that sense, VASIC can go on being developed by the community, incorporating newer LLMs and newer CAS and CAS-based tools, when they become available. The orchestrator policy is modular as well. It can be based on fixed rules, an LLM, a learned classifier, or a combination of them to choose when to retry, change transformation type, switch solvers, or stop—while deciding how best to spend the compute and cost budget, without changing the separation between generation and validation. Future work could test whether such ideas for adaptive orchestration improve overall success or reduce costs.

Looking forward, VASIC's design principles align with a broader trend in LLMs: the necessity of reliable, structured verification. As frontier models become more sophisticated, their failure modes evolve from obvious errors to subtle, highly nuanced hallucinations wrapped in persuasive reasoning. The VASIC framework demonstrates how deterministic tools can be seamlessly combined with generative models to enforce rigorous guardrails without compromising ease-of-use. As frontier AI architectures continue to mature, we anticipate that the external execution and verification loops demonstrated by VASIC will be integrated natively into LLM systems. Native adoption will eventually render external harnesses like VASIC redundant. This evolution would mark a profound success—a shift toward more reliable and trustworthy AI systems for the broader community.

AI use statement. Generative AI systems served as candidate generators for solutions and transformations in the experiments described in this paper. Separately, Generative AI tools assisted with literature retrieval, drafting and editing the manuscript, editing experimental code, and writing code used to generate figures. They were not used to generate or modify the experimental data. Candidate solutions were assessed using the CAS-based validation protocol described in the paper rather than by an AI system. The authors reviewed all AI-assisted text and code, checked the associated claims and citations, and take responsibility for the final content of the work.

Ethics statement. This work does not involve human participants or personal data. Its principal risks concern incorrect mathematical outputs, benchmark contamination, source licensing, proprietary-model terms and version drift, and the computational resources used for evaluation. The system mitigates incorrect-output risk by separating candidate generation from CAS-based acceptance and by reporting unresolved outcomes rather than treating them as validated.

Reproducibility statement. Appendix B documents the algorithms, request configuration, computational limits, software environment, and validation procedure.

REFERENCES

Sina Bagheri Nezhad, Yao Li, and Ameeta Agrawal. SymCode: A neurosymbolic approach to mathematical reasoning via verifiable code generation. In *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 1489–1503. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-eacl.76. URL https://aclanthology.org/2026.findings-eacl.76/.

Rashid Barket, Matthew England, and Jürgen Gerhard. Symbolic integration algorithm selection with machine learning: Lstms vs tree lstms. In *Mathematical Software (ICMS)*, 2024.

Manuel Bronstein. *Symbolic Integration I: Transcendental Functions*, volume 1 of *Algorithms and Computation in Mathematics*. Springer, Berlin, Heidelberg, 2 edition, 2005. ISBN 978-3-540-21493-9. doi: 10.1007/b138171.

Konstantin Chernyshev, Vitaliy Polshkov, Vlad Stepanov, Alex Myasnikov, Ekaterina Artemova, Alexei Miasnikov, and Sergei Tilga. U-MATH: A university-level benchmark for evaluating mathematical skills in large language models. In *Proceedings of the Fourth Workshop on Generation, Evaluation and Metrics (GEM^2)*, pp. 974–1001. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.gem-1.77/.

Meng Fang et al. MathOdyssey: Benchmarking mathematical problem-solving skills in large language models using odyssey math data. *Scientific Data*, 12(1):1392, 2025.

Keith O. Geddes, Stephen R. Czapor, and George Labahn. *Algorithms for Computer Algebra*. Springer, New York, NY, 1992. ISBN 978-0-7923-9259-0. doi: 10.1007/b102438.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=WSrRF5Wy6v.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=IkmD3fKBPQ.

Adam Tauman Kalai, Ofir Nachum, Santosh S. Vempala, and Edwin Zhang. Why language models hallucinate. *arXiv preprint arXiv:2509.04664*, 2025.

Guillaume Lample and François Charton. Deep learning for symbolic mathematics. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1eZYeHFDS.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/aca97732e30bcf1303bc22ac3924fd16-Abstract-Conference.html.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, 2023. URL https://papers.neurips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

MIT Mathematics. MIT integration bee. https://math.mit.edu/~yyao1/integrationbee.html, 2026. Accessed: 2026-07-30.

Michael B. Monagan, Keith O. Geddes, K. Michael Heal, George Labahn, Stefan M. Vorkoetter, James McCarron, and Paul DeMarco. *Maple 10 Programming Guide*. Maplesoft, Waterloo, ON, Canada, 2005.

Joel Moses and Robert H. Risch. The risch algorithms of MACSYMA and SENAC. In *Proceedings of the 1980 MACSYMA Users' Conference*, 1980.

Kimia Noorbakhsh, Modar Sulaiman, Mahdi Sharifi, Kallol Roy, and Pooyan Jamshidi. Pretrained language models are symbolic mathematics solvers too! In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=F7_odJIeQ26.

Erik Panzer. Algorithms for the symbolic integration of hyperlogarithms with applications to feynman integrals. *Computer Physics Communications*, 188:148–166, 2015. doi: 10.1016/j.cpc.2014.10.019.

Albert Rich, Patrick Scheibe, and Nasser Abbasi. Rule-based integration: An extensive system of symbolic integration rules. *Journal of Open Source Software*, 3(32):1073, 2018. doi: 10.21105/joss.01073.

Robert H. Risch. The problem of integration in finite terms. *Transactions of the American Mathematical Society*, 139:167–189, 1969. doi: 10.2307/1995313.

Michael Shalyt, Rotem Elimelech, and Ido Kaminer. ASyMOB: Algebraic symbolic mathematical operations benchmark. *arXiv preprint arXiv:2505.23851*, 2025. URL https://arxiv.org/abs/2505.23851.

Vaibhav Sharma, Abhinav Nagpal, and Muhammed Fatih Balin. SIRD: Symbolic integration rules dataset. In *NeurIPS Workshop on Mathematical Reasoning and AI*, 2023. URL https://neurips.cc/virtual/2023/74981.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=QFjssnKdBI.

Bintao Tang et al. IntegralBench: Benchmarking LLMs with definite integral problems. *arXiv preprint arXiv:2507.21130*, 2025. URL https://arxiv.org/abs/2507.21130.

Quyen Tran, Sutthirut Charoenphon, and Brian T. Howard. CUMath: A benchmark and evaluation framework for LLMs on mathematical reasoning in undergraduate computational math. OpenReview, 2025. URL https://openreview.net/forum?id=3tm37YNMdl.

Mert Unsal, Timon Gehr, and Martin Vechev. AlphaIntegrator: Transformer action search for symbolic integration proofs. *arXiv preprint arXiv:2410.02666*, 2024. URL https://arxiv.org/abs/2410.02666.

Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 3 edition, 2013. doi: 10.1017/CBO9781139856065.

Cornel Ioan Vălean. *(Almost) Impossible Integrals, Sums, and Series*. Springer, 2019.

Cornel Ioan Vălean. *More (Almost) Impossible Integrals, Sums, and Series*. Springer, Cham, 2023.

Wolfram Research, Inc. Mathematica, version 14.3. https://www.wolfram.com/mathematica/, 2025. Wolfram Research, Inc., Champaign, IL; accessed 2026-07-30.

Xin Xu et al. UGMathBench: A diverse and dynamic benchmark for undergraduate-level mathematical reasoning with large language models. *arXiv preprint arXiv:2501.13766*, 2025.

Aojun Zhou, Ke Wang, Zimu Lu, Weikang Shi, Sichun Luo, Zipeng Qin, Shaoqing Lu, Anya Jia, Linqi Song, Mingjie Zhan, and Hongsheng Li. Solving challenging math word problems using GPT-4 code interpreter with code-based self-verification. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=c8McWs4Av0.

A VASIC ALGORITHMS

A.1 GENERAL ORCHESTRATION PROCEDURE

Algorithm 1 General VASIC orchestration procedure

Require: Integral I, solvers $\mathcal{S}$, validator V, orchestrator O

$s \leftarrow \textsc{Initialize}(I)$
while $\textsc{Continue}(O, s)$ **do**
  $(a, S) \leftarrow \textsc{SelectAction}(O, s, \mathcal{S})$
  $p \leftarrow \textsc{Generate}(S, I, s, a)$
  $d \leftarrow \textsc{Assess}(V, I, p, a, S)$
  if $a = \textsc{Solve} \wedge d = \textsc{Pass}$ **then**
    return p
  $s \leftarrow \textsc{Update}(s, a, p, d)$
return UNRESOLVED

In Algorithm 1, the state s holds the input integral together with the history of actions, candidates, and outcomes. At each step the orchestrator picks an action and a solver, and the solver returns a candidate.

ASSESS evaluates each candidate under the chosen acceptance policy, determining whether an answer can be accepted or a transformation can be passed to the downstream solver. An accepted answer ends the loop; otherwise, UPDATE records the candidate and its feedback and makes any accepted transformation available to later solving steps. The loop continues until an answer is accepted or the orchestrator stops.

A.2 DETERMINISTIC ROUTE EVALUATED IN THIS WORK

Algorithm 2 Deterministic VASIC route evaluated in this work

Require: Integral I, solvers $\mathcal{S}$, solver policy π, validator V

$s \leftarrow \textsc{Initialize}(I)$
for $a \in [\textsc{Direct}, \textsc{Adapt}, \textsc{Adapt}]$ **do**
  $S \leftarrow \textsc{SelectSolver}(\pi, \mathcal{S}, a)$
  $F \leftarrow \textsc{Generate}(S, I, s, a)$
  $d \leftarrow \textsc{Assess}(V, I, F, a, S)$
  if $d = \textsc{Pass}$ **then**
    return F
  $s \leftarrow \textsc{Update}(s, a, F, d)$
$\mathcal{A} \leftarrow \emptyset$
for each of the five transformation types τ **do**
  $s_\tau \leftarrow s$
  $a \leftarrow \textsc{Transform}(\tau)$
  for $k = 1$ to 3 **do**
    $S \leftarrow \textsc{SelectSolver}(\pi, \mathcal{S}, a, k)$
    $T \leftarrow \textsc{Generate}(S, I, s_\tau, a)$
    $d \leftarrow \textsc{Assess}(V, I, T, a, S)$
    $s_\tau \leftarrow \textsc{Update}(s_\tau, a, T, d)$
    if $d = \textsc{Pass}$ **then**
      $s_{\text{solve}} \leftarrow s_\tau$
      for $b \in [\textsc{Solve}, \textsc{Adapt}, \textsc{Adapt}]$ **do**
        $S \leftarrow \textsc{SelectSolver}(\pi, \mathcal{S}, b)$
        $F \leftarrow \textsc{Generate}(S, T, s_{\text{solve}}, b)$
        $d \leftarrow \textsc{Assess}(V, I, F, b, S)$
        if $d = \textsc{Pass}$ **then**
          $\mathcal{A} \leftarrow \mathcal{A} \cup \{F\}$
          break
        $s_{\text{solve}} \leftarrow \textsc{Update}(s_{\text{solve}}, b, F, d)$
      break
if $\mathcal{A} \neq \emptyset$ **then**
  return $\mathcal{A}$
return UNRESOLVED

VASIC first attempts the original integral directly, with up to two feedback-guided retries. It assesses every candidate answer against the original integral and returns the first answer that meets its acceptance policy. In this implementation, VASIC accepts explicit CAS answers without further checks and validates LLM-generated answers through CAS.

If none of these attempts yields an accepted answer, VASIC evaluates all five transformation types separately. For each type, the model generates up to three candidate transformations, and the validator supplies feedback after each failed attempt. VASIC sends the first candidate that passes the equivalence check to the solver.

The solver makes one attempt on the accepted transformation and, if necessary, up to two feedback-guided retries. The initial solving attempt and its feedback retries share one context window and a single 20-minute limit. VASIC checks each candidate answer against the original integral. When an answer passes, VASIC adds it to $\mathcal{A}$ and ends the solving sequence for that type. If none of the three attempts passes, VASIC moves to the next type without requesting another transformation of the same type.

The deterministic policy therefore permits at most 33 model calls per problem: three attempts on the original integral, up to three transformation attempts for each of five transformation types, and up to three solving attempts for the first accepted transformation of each type. Actual usage is lower when an earlier route succeeds or an attempt sequence ends early.

In the reported experiments, VASIC evaluates all five types even after one produces an accepted answer, allowing us to compare the types. Other implementations of the general framework may use different transformation menus or stop at the first accepted answer. If no type produces an accepted answer, VASIC returns UNRESOLVED.

Here, s holds the direct-solving and feedback history. Each transformation type starts from this shared history and maintains its own candidate history s_τ. The history s_{solve} also records the solving attempts and feedback for an accepted transformation. F denotes a candidate answer, T a candidate transformed integral, and d the assessment outcome. VASIC enforces the applicable time limits, which may end a sequence before it reaches the maximum attempt count.

B IMPLEMENTATION AND REPRODUCIBILITY DETAILS

B.1 MODEL REQUEST CONFIGURATION

We do not specify top-p, so each provider uses its default value. We also do not specify a random seed because providers do not support a consistent, reproducible seed mechanism across models. A failed request is not silently replaced by a new model sample, ensuring that no approach receives unreported additional attempts.

B.2 PROMPT STRUCTURE AND CALL BUDGETS

The total job limit includes all LLM attempts, parsing, validation, and processing overhead. A timeout was recorded as an operational outcome, rather than evidence that the integral was mathematically unsolvable. The 1,200-second limit applies to each solve job; consequently, a full VASIC run involving multiple jobs may take longer.

Transformation jobs use fixed prompts corresponding to the predefined transformation menu. When a transformation is accepted, the resulting equivalent integral is submitted to the model using the standard integral-solving prompt.

B.3 LOGGING

During the experiments, each request and response was recorded together with its identifier, completion status, token count, and timestamp.

B.4 SOFTWARE AND CAS ENVIRONMENT

Frozen software environment. The configured environment uses Python 3.14.5, SymPy 1.14.0, NumPy 2.4.6, mpmath 1.3.0, wolframclient 1.4.0, and Wolfram Kernel 14.3.0. Each validation call to a CAS backend is capped at 180 seconds, while the complete validation procedure is capped at 365 seconds, including a 5-second cleanup reserve.

B.5 TRANSFORMATION ACCEPTANCE

VASIC first screens each candidate for an unchanged representation, malformed structure, and failure of a preliminary numerical check. For indefinite integrals, it also rejects changes of integration variable that do not include full back-substitution.

The validator then checks mathematical equivalence. For definite transformations, it compares the complete integral values, including changed variables and bounds. For integration by parts, it checks the complete boundary-plus-residual expression. For indefinite transformations, it checks integrand or derivative equivalence rather than a single integral value. In the reported experiments, VASIC treated a conclusive CAS pass as sufficient for transformation acceptance. For definite transformations, numerical agreement between the complete original and transformed integrals could establish acceptance without symbolic confirmation or a second CAS-based numerical check. This agreement provides numerical evidence, not a formal proof.

Only a final PASS admits a transformation to solving. Subsequent diagnostic evaluation does not override that verdict. When constructing the solving dataset, an optional provenance check may

exclude a transformation whose original integral does not match the corresponding original-problem record.

C Additional Results

C.1 Detailed numerical results

Table 1 provides the complete per-model results corresponding to Figure 4. Table 2 provides the final counts for the matched retry comparison in Figure 5. Although both experiments begin with the direct-attempt non-passes, the matched comparison uses separate retry contexts and time limits from the standard VASIC feedback route. Its counts therefore differ from the feedback saves in Table 1.

Table 1: **Validated success rates across twelve LLM systems** on $N = 1{,}445$ problems. Feedback and transformation saves show the additional success rate contributed by each route, in percentage points (pp). Total gain is the increase from direct solving to final VASIC success rate. Values are rounded independently; displayed components may therefore differ slightly from displayed totals.

Model	Direct (%)	Feedback saves (pp)	Transformation saves (pp)	Final VASIC (%)	Total gain (pp)
GPT-5.5 Pro	84.1	3.0	0.8	87.9	+3.8
Gemini 3.1 Pro	85.6	2.8	0.7	89.1	+3.5
DeepSeek V4	75.2	6.7	3.3	85.2	+10.0
DeepSeek Reasoner V3	47.8	6.2	30.0	84.0	+36.2
Gemini 3 Pro	61.4	6.7	8.7	76.8	+15.4
Gemini 2.5 Pro	58.3	3.5	10.3	72.2	+13.9
Kimi K3	55.4	9.6	3.5	68.6	+13.1
Kimi K2.6	31.6	7.1	15.2	53.9	+22.3
Claude Haiku 4.5	35.0	2.5	5.8	43.3	+8.3
GPT-4.1	9.0	6.0	17.2	32.2	+23.2
Mistral Large 2512	14.8	2.2	13.0	30.0	+15.2
Llama 4 Maverick	13.5	0.5	8.0	22.0	+8.5

Table 2: **Feedback versus matched no-feedback retries.** All eligible problems failed the initial direct attempt and were evaluated under both branches. "Solved exclusively by" counts problems solved by one branch but not the other. Net gain is the difference between the branches in distinct problems solved.

		Problems solved		Solved exclusively by		
Model	Eligible problems	Feedback	No feedback	Feedback	No feedback	Net gain
GPT-5.5 Pro	226	121	84	42	5	+37
Gemini 3.1 Pro	94	18	6	13	1	+12
DeepSeek V4	192	74	22	55	3	+52

C.2 Dataset composition

The evaluation set contains 1,104 indefinite and 341 definite integrals. The "Other named sources" category groups problems from mathematical books and expert suggestions.

C.3 CAS baseline

The direct-CAS route evaluated all 1,445 problems. It returned accepted evaluated results for 835 problems (57.8%), timed out on 214, returned an unevaluated integral for 254, and returned an empty result for 142. No runs ended in errors. Each original-integral CAS solve had the same 1,200-second (20-minute) per-job wall-time limit as an LLM solve.

CAS outputs were accepted under the evaluated-return rule and were not passed through the external validator used for LLM-generated answers. The CAS baseline is therefore a contextual reference

Table 3: **Benchmark composition by source family.** Source families are assigned using the retained canonical source labels, with source-candidate records used when the canonical label is missing. Identifiable IntegralBench subcollections are reported separately.

Source family	Problems	Share
Rubi / rule-based corpus	557	38.55%
ASyMOB	344	23.81%
IntegralBench	176	12.18%
U-MATH integral-calculus track	132	9.13%
UMATH/CUMath	106	7.34%
Charlwood collection	45	3.11%
MIT Integration Bee	40	2.77%
Other named university collections	28	1.94%
UGMathBench	9	0.62%
Other named sources	8	0.55%
Total	**1,445**	**100.00%**

Table 4: Strict direct-CAS outcomes on the 1,445-problems evaluation set. The percentages use the full evaluation set as the denominator.

Outcome	Problems	Percentage
Accepted evaluated result	835	57.8%
Timeout	214	14.8%
Unevaluated returned integral	254	17.6%
Empty result	142	9.8%
Total	1,445	100.0%

rather than a directly matched comparison. This strict audit, rather than the inconsistent acceptance and pass fields in the historical phase log, defines the CAS baseline. Transformed CAS runs are maintained separately and are not part of the original-integral CAS baseline. CAS success does not remove an ID from an LLM-specific unrun queue. The CAS ledger records whether a CAS attempted the original integral, whereas an LLM-specific queue records whether a particular LLM has been run under that queue's success rate policy. These are different success rate predicates.

Concrete cross-system example. As a concrete example from IntegralBench (Tang et al., 2025), the problem considers

$$I = \int_0^1 K\left(\frac{x}{4}\right) E(1-x)\,dx, \tag{1}$$

where $K(m)$ and $E(m)$ denote the complete elliptic integrals of the first and second kinds under the parameter convention. Under the configured CAS-only route, Wolfram returned the input unevaluated rather than producing a closed form. The first model attempts produced the outputs in Table 5.

Although the two validated answers have different forms, they are connected by

$$\mathrm{Li}_2\left(\frac{1}{2}\right) - \mathrm{Li}_2\left(-\frac{1}{2}\right) = \frac{\pi^2}{6} - \ln^2 2 - \frac{1}{2}\,\mathrm{Li}_2\left(\frac{1}{4}\right). \tag{2}$$

Numerical evaluation confirms the two validated forms and rejects $\pi \ln 2$, which differs from the integral by approximately 0.00559. This example shows why validation must recognize equivalent closed forms and reject plausible but incorrect expressions.

Complementary indefinite-integration example. An integral from the Rubi test suite (Rich et al., 2018) provides a complementary case in which the first-iteration outputs differ across models. Write the integral as

$$I = \int \frac{N(x)}{D(x)}\,dx, \tag{3}$$

Table 5: Illustrative first-attempt outputs for an IntegralBench problem. This example illustrates behavior behind the aggregate results and is not an independent estimate of model performance.

System	First-attempt output	Outcome
Wolfram CAS	Unevaluated `Integrate[...]` call	Unevaluated
Gemini 3.1 Pro	$\frac{\pi}{4}\left(4-3\ln 3+\frac{\pi^2}{3}-2\ln^2 2-\mathrm{Li}_2\left(\frac{1}{4}\right)\right)$	Pass
GPT-5.5 Pro	$\pi \ln 2$	Fail
DeepSeek V4	$\pi-\frac{3\pi}{4}\ln 3+\frac{\pi}{2}\left[\mathrm{Li}_2\left(\frac{1}{2}\right)-\mathrm{Li}_2\left(-\frac{1}{2}\right)\right]$	Pass

Table 6: System outputs on an illustrative indefinite integral from the Rubi test suite. Pass and fail indicate validation against the original integrand under the stated domain assumptions.

System	Output	Outcome
Wolfram CAS	Evaluated antiderivative matching Equation 4	Pass
Gemini 3.1 Pro Preview	$-x+\ln\left[-1+e^{4/x+x}+\frac{e^x x(19+4\ln x)}{5+\ln x}\right]$	Pass
GPT-5.5 Pro	$\ln\left[(\ln x+5)(e^{x+4/x}-1)+e^x x(4\ln x+19)\right]-\ln(\ln x+5)-x$	Pass
DeepSeek V4 Pro	$\ln\left[\frac{(5+\ln x)(e^{x+4/x}-1)+xe^x(19+4\ln x)}{x(5+\ln x)}\right]$	Fail

where

$$\begin{aligned} N(x) &= 25x^2+e^x(-100e^{4/x}+96x^2) \\ &\quad +\left(10x^2+e^x(-40e^{4/x}+39x^2)\right)\ln x \\ &\quad +\left(x^2+e^x(-4e^{4/x}+4x^2)\right)(\ln x)^2, \\ D(x) &= -25x^2+e^x(25e^{4/x}x^2+95x^3) \\ &\quad +\left(-10x^2+e^x(10e^{4/x}x^2+39x^3)\right)\ln x \\ &\quad +\left(-x^2+e^x(e^{4/x}x^2+4x^3)\right)(\ln x)^2. \end{aligned}$$

For $x>0$, away from singularities of the integrand and candidate expressions, a checked reference antiderivative is

$$F(x)=\ln\left(e^{4/x}-e^{-x}+4x-\frac{x}{5+\ln x}\right). \tag{4}$$

The Wolfram result and the three first-iteration model outputs are shown in Table 6.

To expose the difference, let $B(x)=e^{4/x}-e^{-x}+4x-x/(5+\ln x)$, so that the reference is $F(x)=\ln B(x)$. The Gemini and GPT-5.5 expressions reduce, respectively, to $-x+\ln(e^x B(x))$ and $\ln((5+\ln x)e^x B(x))-\ln(5+\ln x)-x$; their derivatives therefore match the integrand on the admissible real domain. The DeepSeek expression reduces to $\ln(e^x B(x)/x)$, leaving

$$\frac{d}{dx}\left(F_{\mathrm{DeepSeek}}(x)-F(x)\right)=1-\frac{1}{x}\neq 0. \tag{5}$$

Thus, closely related and equally plausible-looking forms can receive different verdicts: two are equivalent antiderivatives, whereas the third contains a subtle normalization error. The comparison is made through derivatives under the declared assumptions rather than through globally branch-insensitive logarithm identities.

Table 7: **Estimated model-inference costs.** Costs are in USD per 100 attempted problems. Additional cost is the difference between full VASIC and direct solving. CAS execution costs are excluded.

Model	Direct	Full VASIC	Additional
GPT-5.5 Pro	56.30	169.70	113.40
Gemini 3.1 Pro	15.90	52.53	36.63
DeepSeek V4	1.74	10.79	9.05
DeepSeek Reasoner V3	1.06	5.68	4.62
Gemini 3 Pro	11.31	38.49	27.18
Gemini 2.5	13.64	99.12	85.48
GPT-4.1	0.3	2.29	1.99
Kimi K3	7.82	48.56	40.74
Kimi K2.6	4.95	27.42	22.47
Claude Haiku 4.5	0.56	3.66	3.1
Mistral Large 2512	0.13	0.82	0.69
Llama 4 Maverick	0.028	0.169	0.141

Both branches of the feedback comparison run on the problems that did not pass the direct attempt, with the same model, decoding settings, and validator configuration. Each branch gets up to two retries, each in its own context window with a 20-minute limit. The retry budget is therefore identical across branches, although token consumption may differ. In standard VASIC, by contrast, the direct attempt and its two feedback retries share one context window and a single 20-minute limit.

C.4 Inference costs

Table 7 reports estimated model-inference costs for direct solving and full VASIC, including unsuccessful attempts and excluding CAS execution. The additional cost varies widely across models, from $0.141 per 100 problems for Llama 4 Maverick to $113.40 for GPT-5.5 Pro. Higher spending does not necessarily yield larger gains in validated success rate: DeepSeek Reasoner V3 gains 36.2 percentage points at an additional cost of $4.62 per 100 problems, while GPT-5.5 Pro gains 3.8 points at a substantially higher cost. These results suggest that model choice should account for both direct-solving performance and the cost of solving additional solutions.

C.5 Available timing measurements

We report timing for the retry stage separately because the full VASIC pipeline was executed in separate experimental batches. Among problems with complete timing records for both branches, median elapsed time for retries with versus without feedback was 714.8 versus 456.5 seconds for Gemini 3.1 Pro, 1,837.6 versus 960.0 seconds for DeepSeek V4, and 137.7 versus 146.4 seconds for the GPT-family runs. These measurements cover up to two retries and exclude the shared initial attempt and the transformation routes.

As a separate operational reference, we measured standalone CAS execution on 758 problems, running SymPy and Wolfram independently with a 60-second timeout. Across all outcomes, SymPy had a median runtime of 1.54 seconds and timed out on 111 problems (14.6%); its 95th-percentile runtime was 60.03 seconds. Wolfram had a median runtime of 0.87 seconds and timed out on 23 problems (3.0%); its 95th-percentile runtime was 51.46 seconds. These are standalone CAS runtime measurements rather than timings of validator calls within VASIC.

C.6 Validator evidence

The validator provides an acceptance decision, not a formal proof. The stress cases below illustrate how symbolic checks, numerical evidence, and backend limitations affect individual decisions.

We drew 100 cases at random from each of four pools. The correct answers had already been verified by both CAS backends, and the incorrect answers were sign-flip, factor-of-two, and dropped-term variants of them. For the two transformation pools, one CAS had earlier labeled each case as equivalent or non-equivalent, and we re-tested those cases with the other CAS. Each transformation pool contained 20 cases from each of the five transformation types.

Table 8: Validator outcomes on 400 sampled cases with expected labels assigned before testing.

Case	Tested	Pass	Fail	Inconclusive
Correct answers	100	97	0	3
Incorrect answers	100	0	96	4
Equivalent transformations	100	96	0	4
Non-equivalent transformations	100	0	95	5

Table 9: Independent-CAS rechecks of accepted candidate records. Symbolic and numerical refer to the original acceptance evidence.

Original acceptance	Rechecked	Confirmed	Contradicted	Inconclusive
Answer, symbolic	291	235	0	56
Answer, numerical	283	162	1	120
Transformation, symbolic	270	223	1	46
Transformation, numerical	193	174	1	18
Total	1,037	794	3	240

No incorrect answer or non-equivalent transformation received PASS. The validator returned INCONCLUSIVE on 16 of the 400 cases. These numbers apply only to the sampled cases, and the transformation labels come from a single CAS's earlier decisions, not from independent mathematical adjudication.

We drew a stratified random sample from the ledger of accepted results, grouping records by type (answer or transformation), acceptance mode (symbolic or numerical) and the authorising CAS, and sampling within each group. A record was eligible only if exactly one identifiable CAS had accepted it and its candidate and original integral could be identified, so records supported by both CAS were excluded. We fixed the sample before rechecking the 1,037 selected records with the other CAS. Repeated acceptances of the same candidate for the same integral counted once; different candidates for one integral counted separately. We balanced each stratum by originating CAS as far as the eligible records allowed, so pooled counts do not estimate ledger-wide rates. Table 9 groups the results by the original acceptance evidence.

We examined each of the three contradictions by hand. One numerically accepted answer was wrong. In one transformation, the original integral and its integration-by-parts rewrite both equal 113, but the rechecking CAS missed kinks in the integrand. The third transformation is mathematically equivalent, though its form raises a separate question about what the transformation policy should admit. So only one of the three contradictions turned out to be an incorrect acceptance.

C.7 VALIDATOR STRESS CASES

Dependence on CAS capabilities can become a validation bottleneck: a correct candidate may remain inconclusive when the available backends cannot establish the identity within their limits. This strictness preserves the trust boundary: unresolved evidence cannot justify accepting a candidate as correct. Appendix C.7 illustrates how numerical evidence can support a candidate while symbolic uncertainty remains unresolved.

The validator is designed to combine the strengths of different CAS backends rather than to replace or improve any one of them. It uses symbolic checks when they succeed, numerical evaluation as additional validation or fallback evidence, and independent-backend confirmation when one system remains inconclusive. Two reproducible cases illustrate the value of this combination.

First, for $x^2\big(3x^3\cosh(2x^3)+\sinh(2x^3)\big)$, an antiderivative written with exponentials is mathematically valid, but standard SymPy simplification does not reduce its derivative residual to zero. A guarded rewrite into a common exponential representation allows the SymPy route to establish the identity exactly; numerical evaluation supplies an additional check. The production route terminates on this SymPy pass, while a separate Wolfram-only audit independently validates all three recorded candidate forms. The point is not that VASIC is generally better than SymPy, but that con-

Integral

$$\int_{-4}^{4} |x^3 - x|\, dx$$

candidate A - × rejected

$$\int_{-4}^{4} |\sin^3 x - \sin x|\, dx$$

The DeepSeek candidate replaces x by $\sin x$ without a valid change of variables, so the equivalence gate rejects it and no transformed solve follows.

candidate B - ✓ accepted

$$\int_{-4}^{4} |x|\,|x^2 - 1|\, dx$$

The factorization passes the transformation check. The downstream solver returns 113, which passes numerical validation against the original integral.

Reference audit passed; reference value 113.

Figure 6: Contrasting transformation candidates for the same integral. The candidates were recorded in separate trajectories; the comparison illustrates validator-gated selection rather than a single sequential run.

trolled escalation and a second backend can reduce false rejections caused by representation-specific limitations.

Second, for $\int \mathrm{ArcTan}(x)/(x^2\sqrt{1+x^2})\, dx$ under $x > 0$, SymPy's symbolic strategies remain inconclusive for a valid branch-sensitive candidate, while deterministic numerical differentiation gives a maximum residual of 2.66×10^{-15} over 100 usable points. This is retained as positive numerical evidence, but the SymPy-only consensus remains inconclusive because a single numerical channel is not sufficient for high-assurance acceptance. In the full configuration, Wolfram independently returns a symbolic-plus-numerical pass, and the combined consensus accepts the candidate. Thus, numerical checking has two useful roles in VASIC: it can strengthen a symbolic result, and it can preserve evidence when symbolic checking is unresolved without silently turning that uncertainty into a pass.

C.8 Robustness and failure analysis

Figure 6 gives a concrete example of the transformation gate rejecting a plausible-looking but non-equivalent representation. The accepted and rejected candidates come from separate transformation trajectories for the same problem and are shown together only to make the acceptance boundary visible.

D Examples of the VASIC Process

The following examples show how validator decisions affect individual trajectories. They were selected from records whose reference answers passed an independent audit and whose displayed paths contain neither timeouts nor inconclusive events. They are examples of validator-gated behavior, not a random sample or an estimate of the average success-rate gain. The matched experiment in Section 4.4, rather than these selected traces, supports the claim that validator-derived feedback improves success rate.

D.1 Validator-guided retry

Figure 7 shows a failed candidate followed by an accepted feedback-guided retry. The stored record contains the candidates and validator verdicts but not the exact feedback wording, so the box does not reconstruct the missing message.

Integral

$$\int_0^1 \frac{\log\left(\frac{x+1}{1-x}\right)\log\left(\frac{\sqrt{1-x^2}+1}{1-\sqrt{1-x^2}}\right)}{x}\,dx$$

Attempt 1 - × rejected Gemini 3.1 Pro

$$4\pi G - 11\zeta(3) + \pi^2 \log 2$$

Recorded verdict: fail. An independent Wolfram audit confirms that the candidate is mathematically incorrect rather than merely rejected for formatting or policy reasons.

$\Downarrow$ validator-guided retry

Attempt 2 - ✓ accepted Gemini 3.1 Pro

$$\pi^2 \log 2$$

Recorded verdict: numerical pass.

Reference audit passed; reference value $\pi^2 \log 2$.

Figure 7: Illustrative validator-guided retry. Here G denotes Catalan's constant. The trace demonstrates validator-gated selection; the matched experiment provides evidence for the effect of feedback.

D.2 VALIDATED TRANSFORMATION

Figure 8 shows how the transformation gate screens representation candidates before they reach the solver. We distinguish structural and formatting rejections from mathematical non-equivalence verdicts.

Original integral

$$\int_{-\infty}^{\infty} e^{-2x^2-5x-3}\,dx$$

Valid substitution

$$-2x^2-5x-3=\frac{1}{8}-2\left(x+\frac{5}{4}\right)^2, \qquad x+\frac{5}{4}=\frac{\tan u}{\sqrt{2}}.$$

Transformation attempt 1 - × rejected structure
The candidate places the Jacobian outside the top-level `Integrate` expression. VASIC rejects the structure without checking mathematical equivalence.

Transformation attempt 2 - × rejected format
The candidate returns an unevaluated integral-plus-replacement-rule program rather than a self-contained transformed integral. VASIC rejects the format without performing a CAS equivalence check. The candidate also uses $e^{-1/8}$ instead of $e^{1/8}$.

Transformation attempt 3 - ✓ accepted

$$\frac{e^{1/8}}{\sqrt{2}}\int_{-\pi/2}^{\pi/2} e^{-\tan^2 u}\sec^2 u\,du$$

Wolfram's whole-integral numerical comparison returned PASS, with an absolute difference of approximately 1.63×10^{-11}. The acceptance policy used in this run admitted the transformation on that evidence alone.

⇓ solve accepted representation

Final answer - ✓ accepted

$$e^{1/8}\sqrt{\frac{\pi}{2}}$$

Gemini 3.1 Pro returns the result, which passes numerical validation against the original integral.

Reference audit passed; reference value $e^{1/8}\sqrt{\pi/2}$.

Figure 8: Illustrative validated-transformation route. VASIC rejects two candidates for structural or formatting reasons and admits the third after a whole-integral equivalence check. Gemini 3.1 Pro generated the transformation and solved it. This separate illustrative example does not contribute to the aggregate results.