MathSticks: A Benchmark for Visual Symbolic Compositional Reasoning with Matchstick Puzzles

Yuheng Ji[1,2,3,*], Huajie Tan[3,4,*], Cheng Chi[3,*], Yijie Xu[3,5], Yuting Zhao[1,2], Enshen Zhou[3,6],
Huaihai Lyu[1,2,3], Pengwei Wang[3], Zhongyuan Wang[3], Shanghang Zhang[3,4,✉], Xiaolong Zheng[1,2,✉]
[1] Institute of Automation, Chinese Academy of Sciences [2] School of Artificial Intelligence, University of Chinese Academy of Sciences [3] Beijing Academy of Artificial Intelligence
[4] Peking University [5] The University of Sydney [6] Beihang University

Abstract

We introduce MATHSTICKS, a benchmark for Visual Symbolic Compositional Reasoning (VSCR), which unifies visual perception, symbolic manipulation, and arithmetic consistency. Each task presents an incorrect matchstick equation that must be corrected by moving one or two sticks under strict conservation rules. The benchmark includes both text-guided and purely visual settings, systematically covering digit scale, move complexity, solution multiplicity, and operator variation, with a full dataset of 1.4M generated instances. For standardized evaluation, we release a balanced 400-item benchmark set. Evaluations of 14 vision–language models reveal substantial limitations: closed-source models succeed only on simple cases, open-source models fail in the visual regime, while humans exceed 90% accuracy. These findings establish MATHSTICKS as a rigorous testbed for advancing compositional reasoning across vision and symbols. Our code and dataset are publicly available at `https://github.com/Yuheng2000/MathSticks`.

1 Introduction

Existing tasks in vision and reasoning often tackle only one dimension, such as visual perception, reasoning, or arithmetic computation, without forming a unified challenge. To bridge this gap, we introduce Visual Symbolic Compositional Reasoning (VSCR), which unifies all three by linking visual perception with symbolic manipulation. VSCR requires models to recognize structured elements, plan transformations under explicit constraints, and verify arithmetic consistency, reflecting core aspects of everyday cognition. Yet current vision–language models (VLMs) struggle with symbol-level understanding and constrained edits, while existing benchmarks remain limited. As summarized in Tab. 1, they lack: *(i)* constrained symbolic manipulation with solvability guarantees; *(ii)* fine-grained control of task settings and difficulty; and *(iii)* evaluation protocols isolating pure-visual reasoning.

To close these gaps, we present MATHSTICKS, a benchmark built on matchstick arithmetic puzzles. Each task is an incorrect equation corrected by moving one or two sticks, enforcing both stick conservation and arithmetic consistency. Two regimes are defined: a text-prompted setting, where the equation string is provided, and a pure-visual setting, where only the rendered puzzle is shown (Fig. 1). This separation enables diagnosis of symbolic reasoning versus visual parsing. The pipeline systematically enumerates digit-level operations, validates solvability, and generates 1.4M instances, from which a balanced evaluation set of 400 items is released. The benchmark spans digit scale (Levels 1–4), move

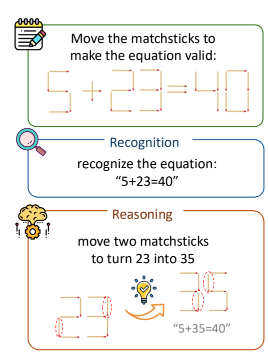


Figure 1: Overview of the MATHSTICKS task.

* Equal contribution. ✉ Corresponding authors.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

Table 1: Comparison of representative benchmarks across five key dimensions. *Partial* denotes limited coverage in specific sub-tasks or without a unified protocol.

Benchmark	Valid editing	Executable check	Difficulty slices	Symbolic grounding	Scale (≈1M)
CLEVR [1]	✗	✓	✗	*partial*	✗
CLEVR-Change [2]	✗	✗	✗	✗	✗
MathVista [3]	✗	*partial*	✗	*partial*	✗
AI2D [4]	✗	✗	✗	*partial*	✗
ChartQA [5]	✗	*partial*	✗	*partial*	✗
TRANCE [6]	✓	✗	*partial*	*partial*	✗
VisualTrans [7]	✓	✗	*partial*	✗	✗
MathSticks (Ours)	✓	✓	✓	✓	✓

complexity, solution multiplicity, and operator flipping, while avoiding extreme cases to ensure human accessibility.

We evaluate 14 representative VLMs, including closed-source models (e.g., o3, Gemini 2.5 Pro) and open-source families (Qwen2.5-VL, InternVL). Results show a clear capability gap: closed models solve simple puzzles but falter on multi-move and operator-flip cases, open-source models score near zero in the pure-visual regime, and humans consistently exceed 90% accuracy. This establishes MATHSTICKS as a controlled, diagnostic testbed for advancing VSCR.

2 MathSticks

Task Definition. Each puzzle is an equation $x = [a, \oplus, b, =, c], \oplus \in \{+, -\}$, rendered in matchstick-based seven-segment digits. The input is always an *invalid* equation. The objective is to relocate $k \in \{1, 2\}$ sticks, without insertion or deletion, such that the corrected form satisfies $a \oplus b = c$. This formulation integrates three components: visual perception of digits and operators, planning of valid symbolic transformations under strict constraints, and verification of arithmetic correctness.

For precise evaluation, every matchstick position is indexed at the segment level (e.g., `A0-A6`, `B0-B6`). During evaluation, VLMs are required to output edits in a canonical format such as `Move(A0, C3)` for single moves or `Move(A0, C3), Move(E1, F4)` for two moves. This ensures that predictions are machine-parsable, unambiguous, and directly verifiable against ground-truth solutions. Detailed visual conventions are provided in Appendix B.

Categorization. To support fine-grained diagnosis, we annotate each instance along four axes: (i) digit scale (Levels 1–4, from single- to two-digit operands), (ii) move complexity (1- vs. 2-move solvability), (iii) solution multiplicity (unique vs. multiple), and (iv) operator flipping (sign-changed vs. preserved). These factors yield seven diagnostic categories, with precise definitions and illustrations deferred to the Appendix E.2.

We acknowledge that stratifying difficulty solely by digit scale is a simplification. The true cognitive difficulty can also be influenced by factors such as the visual ambiguity of potential moves or the complexity of the required arithmetic. For instance, a single-digit puzzle requiring an operator flip might be more challenging than a two-digit puzzle with a straightforward digit edit. A deeper, empirically-driven analysis of difficulty metrics remains a valuable direction for future work.

Dataset Construction. We design a two-stage generation pipeline that guarantees both completeness in the symbolic search space and fidelity in the rendered visual stimuli. (1) *Symbolic enumeration*: each equation is represented as a 7-slot state and exhaustively explored under legal one- and two-stick moves with stick conservation. Candidate edits are filtered by arithmetic validation, deduplicated across move types, and labeled with diagnostic attributes (digit scale, move complexity, solution multiplicity, operator flipping). (2) *Visual rendering*: each symbolic state is deterministically mapped to an image via a template library of seven-segment digits and operator slots, ensuring one-to-one correspondence between symbolic edits and visible stick relocations. This construction yields $\sim$1.4M solvable instances with large-scale coverage and structured metadata. Algorithmic details and pseudocode are provided in Appendix C.

Dataset Statistics. The benchmark comprises 1,411,388 validated solvable instances. Difficulty is highly skewed toward Level 4 (79.07%), with Levels 1–3 contributing 0.11%, 1.31%, and 19.51%, respectively. Most instances require two-stick edits (82.01%), while one-stick solutions are rare

(4.18%); the remainder admit both one- and two-stick corrections (13.81%). Regarding solution multiplicity, 43.12% of instances have a unique correction, whereas 56.88% admit multiple valid corrections. For standardized and reproducible evaluation, we constructed the MATHSTICKS benchmark, a compact 400-item test set. This benchmark set is stratified evenly across Levels 1–4 (100 items per level) to ensure a balanced assessment of model capabilities across different difficulties. All experiments reported in this paper, for both VLMs and humans, are conducted on this 400-item benchmark. More detailed statistics are provided in Appendix D.

3 Experiments

3.1 Experimental Setup

Setup and Protocol. We evaluate models on the MATHSTICKS benchmark under two input regimes: (i) *text-prompted*, where the symbolic equation string is explicitly provided; and (ii) *pure-visual*, where only the matchstick image is given, requiring OCR and structural parsing before subsequent reasoning. Each instance is further categorized into four difficulty levels (L1–L4), ranging from equations composed entirely of single-digit operands to cases involving one or more two-digit numbers. This hierarchy captures progressively greater visual and structural complexity. Additional dimensions such as operation complexity, solution multiplicity, and operator flipping are included for comprehensive diagnosis, with detailed breakdowns deferred to Appendix E.2. The primary evaluation metric is accuracy (%), computed per level and regime.

Human Evaluation. To provide a reference ceiling, we recruited three adult participants who independently solved the full 400-item evaluation set under the *pure-visual regime*, where only rendered matchstick equations were shown. Because digits and operators are visually unambiguous, the same accuracies apply to the text-prompted regime. On average, participants exceeded 90% accuracy with about one minute per problem, confirming the solvability of all tasks and highlighting the gap to current models. Detailed per-level results and solution times are reported in Appendix E.3.

Evaluated Models. We include both closed and open-source VLMs. For *closed models*, we evaluate o3 [8], Gemini 2.5 Pro [9], Gemini 2.5 Flash [9], GPT-o4-mini [8], Seed-1.6-Thinking [10], Seed 1.6 [10], Claude Sonnet 4 [11], and GPT-4o [12]. For *open-source models*, we cover two representative families: Qwen2.5-VL (7B, 32B, 72B) [13]and InternVL3 (8B, 38B, 78B) [14]. In addition to the aggregate results reported here, we provide a fine-grained breakdown across difficulty dimensions in Appendix E.2.

3.2 Performance Analysis

The benchmark results in Tab. 2 reveal systematic differences across model families and input regimes, as well as between human and model performance. These comparisons provide a structured view of how current VLMs engage with symbolic constraints, and where the main performance gaps emerge.

Closed-source models substantially outperform open-source systems. Proprietary models attain markedly higher accuracies, whereas open-source baselines remain at or near chance level across both regimes. The effect is large and consistent across difficulty levels, underscoring the challenge of structured reasoning for today's open-source pipelines. As we detail in Sec. 3.3, this widespread failure stems not just from incorrect reasoning but from a more fundamental inability to adhere to the task's symbolic rules and output structure.

Reasoning-optimized variants outperform their standard counterparts. Within the closed-source group, models advertised as reasoning-enhanced achieve higher accuracy: o3 reaches 60.00% compared with 0.00% for GPT-4o; Gemini-2.5-Pro achieves 45.25% compared with 26.50% for Gemini-2.5-Flash; Seed-1.6-Thinking reaches 21.50% compared with 3.25% for its base model. These observations point to the importance of stepwise and structured reasoning mechanisms for handling compositional edits.

o3 exhibits the strongest overall performance. It attains the highest average accuracy (49.25%) across regimes, followed by Gemini-2.5-Pro (33.88%). Its "thinking with image" paradigm, which integrates visual parsing with multi-step symbolic reasoning, appears particularly effective for puzzles requiring both perceptual discrimination and arithmetic verification.

Table 2: **Results on the MathSticks Reasoning Benchmark.** We report accuracy (%) across difficulty levels under two regimes (w/ and w/o text prompt). The best results are in **bold** and the second-best are underlined. † Human evaluation was conducted only under the pure-visual regime. Since the rendered digits and operators are visually unambiguous, participants consistently recognized the equations without error; thus, providing the equation string does not alter the task.

Model	MathSticks (w/ text prompt)					MathSticks (w/o text prompt)				
	L1	L2	L3	L4	AVG	L1	L2	L3	L4	AVG
Closed Models										
o3-250416	73.00	56.00	56.00	55.00	**60.00**	69.00	37.00	30.00	18.00	**38.50**
Gemini-2.5-Pro-250506	67.00	41.00	35.00	38.00	45.25	42.00	19.00	21.00	8.00	22.50
Gemini-2.5-Flash-250520	53.00	19.00	19.00	15.00	26.50	22.00	7.00	5.00	0.00	8.50
GPT-o4-mini-250416	47.00	19.00	21.00	12.00	24.75	30.00	7.00	3.00	2.00	10.50
Seed-1.6-Thinking-250615	41.00	14.00	18.00	13.00	21.50	3.00	0.00	0.00	1.00	1.00
Seed-1.6-250615	8.00	1.00	1.00	3.00	3.25	2.00	0.00	0.00	0.00	0.50
Claude Sonnet 4	7.00	0.00	2.00	0.00	2.25	0.00	0.00	0.00	0.00	0.00
GPT-4o-241120	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Open-Source Models										
Qwen2.5-VL-7B-Instruct	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Qwen2.5-VL-32B-Instruct	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Qwen2.5-VL-72B-Instruct	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
InternVL3-8B	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
InternVL3-38B	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
InternVL3-78B	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Human Performance										
Human†	95.00	98.33	85.83	87.72	91.72 ± 6.13	95.00	98.33	85.83	87.72	91.72 ± 6.13

Explicit textual input yields consistent gains. Providing the equation string improves accuracy for all models (e.g., o3 improves from 38.50% without text to 60.00% with text). Without text, models must additionally perform OCR-like recognition of digits and operators, making perception a salient bottleneck for end-to-end performance.

Humans maintain a clear advantage. Participants achieve an average accuracy of 91.72%, well above the best model's 49.25%. This confirms that the benchmark is readily solvable for humans, while exposing persistent limitations of current VLMs in integrating perception and reasoning.

Overall, these results delineate a consistent pattern: while closed-source reasoning-enhanced models make tangible progress, a wide gap remains to human-level strategies, which involve reliable multi-stage reasoning that current systems cannot yet replicate.

3.3 Error Analysis

We identify five primary error types observed in model outputs. Each reflects a distinct weakness in the perception–reasoning–output pipeline, illustrated with representative cases.

Perception errors. Models occasionally misread digits or operators in the seven-segment display. For example, a common failure is confusing visually similar digits such as "5" and "6," or overlooking the short vertical bar required to distinguish "1" from "7." These errors reveal insufficient robustness in low-level visual parsing when symbolic cues are absent.

Edit-planning errors. Some solutions violate the basic rules of matchstick manipulation, such as proposing a move from a non-existent segment or implicitly adding an extra stick. These mistakes suggest that models often lack an internalized notion of structural constraints governing legal edits.

Arithmetic-verification errors. In certain cases, models generate edits that produce an equation which is visually valid but numerically incorrect. For instance, transforming $9 - 3 = 5$ into $9 - 2 = 5$ satisfies the stick-move requirement but results in an arithmetically false statement. Such cases show that models may stop at surface plausibility without executing a full arithmetic check.

Operator-handling errors. Another weakness arises in tasks requiring operator changes. For example, when converting $7 + 2 = 9$ into a valid form, some models attempt digit-level edits while leaving the operator unchanged, or misplace the segment needed to form a minus. Although less frequent than other error types, such mistakes directly undermine equation validity.

Output-format errors. Finally, several models deviate from the required structured output format (e.g., `Move(A0, C3)`), instead producing free-form natural language such as "move the top stick from the 9 to make it a 3." These outputs prevent automatic evaluation even when the underlying reasoning is partially correct.

Taken together, these error types align closely with the difficulty dimensions summarized in Appendix E.2—for instance, edit-planning errors are prevalent in two-stick puzzles, operator-handling errors concentrate in flip cases, and perception errors intensify under pure-visual inputs.

Systemic Failures in Open-Source Models. A particularly striking finding of our study is the near-total failure of open-source models across all tasks, a result that goes beyond simple performance gaps. While stronger closed-source models typically fail due to one of the aforementioned error types in isolation, our analysis reveals that open-source models often exhibit a *cascade of these failures simultaneously*. For example, a representative response might correctly perform OCR on the input equation but then proceed to propose an edit that violates conservation rules, follows a flawed arithmetic chain, and is ultimately presented in an incorrect output format, as detailed with qualitative examples in Appendix E.2. This pattern of compound errors suggests their near-zero performance stems not from isolated weaknesses in perception or reasoning, but from a more fundamental inability to internalize the task's constrained, procedural, and symbolic nature.

4 Conclusion

We introduced MATHSTICKS, a benchmark targeting visual symbolic compositional reasoning (VSCR) through matchstick arithmetic puzzles. The benchmark enforces solvability guarantees, provides stratified difficulty slices, and supports both text-prompted and pure-visual evaluation, enabling controlled and diagnostic assessment. Experiments with 14 VLMs reveal that even strong closed-source systems struggle with multi-step edits and operator flips, while open-source models collapse in the pure-visual regime. Human participants, by contrast, exceed 90% accuracy. These results highlight a substantial capability gap and establish MATHSTICKS as a compact yet challenging testbed for future progress in perception–symbol reasoning.

Limitations and Future Work. We acknowledge that MATHSTICKS serves as a controlled testbed, intentionally designed to isolate core symbolic and arithmetic reasoning mechanisms from the complexities of unconstrained visual perception. A primary limitation therefore lies in its synthetic nature. Evaluating how the compositional skills diagnosed by our benchmark generalize to *in-the-wild* scenarios, which present diverse object styles, cluttered backgrounds, and varied perspectives, remains a critical open question for future research. A related limitation pertains to our difficulty stratification. While based on the intuitive factor of digit scale, this metric does not encompass the full spectrum of cognitive complexity inherent in more naturalistic problems. Future work should therefore aim to bridge this gap, both by extending the benchmark towards more varied and realistic visual settings and by developing more granular, empirically-grounded difficulty metrics.

Acknowledgments

This work was supported by the Natural Science Foundation of China under Grant Nos. 72225011, 72434005, and 62476011, as well as by the National Science and Technology Major Project (No. 2022ZD0117800).

References

[1] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2901–2910, 2017.

[2] Dong Huk Park, Trevor Darrell, and Anna Rohrbach. Robust change captioning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4624–4633, 2019.

[3] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical

reasoning of foundation models in visual contexts. In *The Twelfth International Conference on Learning Representations*.

[4] Tuomo Hiippala, Malihe Alikhani, Jonas Haverinen, Timo Kalliokoski, Evanfiya Logacheva, Serafina Orekhova, Aino Tuomainen, Matthew Stone, and John A Bateman. Ai2d-rst: a multimodal corpus of 1000 primary school science diagrams. *Language Resources and Evaluation*, 55(3):661–688, 2021.

[5] Ahmed Masry, Xuan Long Do, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2263–2279, 2022.

[6] Xin Hong, Yanyan Lan, Liang Pang, Jiafeng Guo, and Xueqi Cheng. Transformation driven visual reasoning. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 6903–6912, 2021.

[7] Yuheng Ji, Yipu Wang, Yuyang Liu, Xiaoshuai Hao, Yue Liu, Yuting Zhao, Huaihai Lyu, and Xiaolong Zheng. Visualtrans: A benchmark for real-world visual transformation reasoning. *arXiv preprint arXiv:2508.04043*, 2025.

[8] OpenAI. Openai o3 and o4-mini system card. `https://openai.com/index/introducing-o3-and-o4-mini/`, 2025. Accessed: 2025-04-16.

[9] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Ori Ram, Dan Zhang, Evan Rosen, Luke Marris, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

[10] ByteDance Seed, Jiaze Chen, Tiantian Fan, Xin Liu, Lingjun Liu, Zhiqi Lin, Mingxuan Wang, Chengyi Wang, Xiangpeng Wei, Wenyuan Xu, et al. Seed1. 5-thinking: Advancing superb reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.13914*, 2025.

[11] Anthropic. Claude 3.7 sonnet and claude code. `https://www.anthropic.com/news/claude-3-7-sonnet`, 2025. Accessed: 2025-02-25.

[12] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[13] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[14] Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*, 2024.

[15] Shuanghao Bai, Wenxuan Song, Jiayi Chen, Yuheng Ji, Zhide Zhong, Jin Yang, Han Zhao, Wanqi Zhou, Wei Zhao, Zhe Li, et al. Towards a unified understanding of robot manipulation: A comprehensive survey. *arXiv preprint arXiv:2510.10903*, 2025.

[16] Jincai Huang, Yongjun Xu, Qi Wang, Qi Cheems Wang, Xingxing Liang, Fei Wang, Zhao Zhang, Wei Wei, Boxuan Zhang, Libo Huang, et al. Foundation models and intelligent decision-making: Progress, challenges, and perspectives. *The Innovation*, 2025.

[17] Chunjie Zhang, Chuankai Liu, Shaohua Duan, Xiaolong Zheng, Tianyi Yu, and Jitao Zhang. Embodied cognitive intelligence guided moon sample collection. *The Innovation*, 2025.

[18] Yuting Zhao, Yuheng Ji, Xiaoshuai Hao, and Shuxiao Li. Fastrsr: Efficient and accurate road surface reconstruction from bird's eye view. *arXiv preprint arXiv:2504.09535*, 2025.

[19] Huaihai Lyu, Hantao Yao, and Changsheng Xu. Multiple local prompts distillation for domain generalization. *IEEE Transactions on Multimedia*, 2025.

[20] Enshen Zhou, Qi Su, Cheng Chi, Zhizheng Zhang, Zhongyuan Wang, Tiejun Huang, Lu Sheng, and He Wang. Code-as-monitor: Constraint-aware visual programming for reactive and proactive robotic failure detection. *arXiv preprint arXiv:2412.04455*, 2024.

[21] Enshen Zhou, Jingkun An, Cheng Chi, Yi Han, Shanyu Rong, Chi Zhang, Pengwei Wang, Zhongyuan Wang, Tiejun Huang, Lu Sheng, et al. Roborefer: Towards spatial referring with reasoning in vision-language models for robotics. *arXiv preprint arXiv:2506.04308*, 2025.

[22] Huaihai Lyu, Chaofan Chen, Yuheng Ji, and Changsheng Xu. Egoprompt: Prompt pool learning for egocentric action recognition. *arXiv preprint arXiv:2508.03266*, 2025.

[23] Songran Bai, Yuheng Ji, Yue Liu, Xingwei Zhang, Xiaolong Zheng, and Daniel Dajun Zeng. Alleviating performance disparity in adversarial spatiotemporal graph learning under zero-inflated distribution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 11436–11444, 2025.

[24] Zhe Li, Laurence T Yang, Xin Nie, BoCheng Ren, and Xianjun Deng. Enhancing sentence representation with visually-supervised multimodal pre-training. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 5686–5695, 2023.

[25] Zhe Li, Zhangyang Gao, Cheng Tan, Stan Z Li, and Laurence T Yang. General point model with autoencoding and autoregressive. *arXiv preprint arXiv:2310.16861*, 2023.

[26] Zhe Li, Weihao Yuan, Yisheng He, Lingteng Qiu, Shenhao Zhu, Xiaodong Gu, Weichao Shen, Yuan Dong, Zilong Dong, and Laurence T Yang. Lamp: Language-motion pretraining for motion generation, retrieval, and captioning. *arXiv preprint arXiv:2410.07093*, 2024.

[27] Zhe Li, Yisheng He, Lei Zhong, Weichao Shen, Qi Zuo, Lingteng Qiu, Zilong Dong, Laurence Tianruo Yang, and Weihao Yuan. Mulsmo: Multimodal stylized motion generation by bidirectional control flow. *arXiv preprint arXiv:2412.09901*, 2024.

[28] Zhe Li, Laurence T Yang, Bocheng Ren, Xin Nie, Zhangyang Gao, Cheng Tan, and Stan Z Li. Mlip: Enhancing medical visual representation with divergence encoder and knowledge-guided contrastive learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11704–11714, 2024.

[29] Mengzhen Liu, Mengyu Wang, Henghui Ding, Yilong Xu, Yao Zhao, and Yunchao Wei. Segment anything with precise interaction. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 3790–3799, 2024.

[30] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[31] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, et al. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

[32] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

[33] Huajie Tan, Yuheng Ji, Xiaoshuai Hao, Minglan Lin, Pengwei Wang, Zhongyuan Wang, and Shanghang Zhang. Reason-rft: Reinforcement fine-tuning for visual reasoning. *arXiv preprint arXiv:2503.20752*, 2025.

[34] Yuheng Ji, Huajie Tan, Jiayu Shi, Xiaoshuai Hao, Yuan Zhang, Hengyuan Zhang, Pengwei Wang, Mengdi Zhao, Yao Mu, Pengju An, et al. Robobrain: A unified brain model for robotic manipulation from abstract to concrete. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 1724–1734, 2025.

[35] BAAI RoboBrain Team, Mingyu Cao, Huajie Tan, Yuheng Ji, Minglan Lin, Zhiyu Li, Zhou Cao, Pengwei Wang, Enshen Zhou, Yi Han, et al. Robobrain 2.0 technical report. *arXiv preprint arXiv:2507.02029*, 2025.

[36] Huajie Tan, Xiaoshuai Hao, Cheng Chi, Minglan Lin, Yaoxu Lyu, Mingyu Cao, Dong Liang, Zhuo Chen, Mengsi Lyu, Cheng Peng, et al. Roboos: A hierarchical embodied framework for cross-embodiment and multi-agent collaboration. *arXiv preprint arXiv:2505.03673*, 2025.

[37] Zirui Song, Guangxian Ouyang, Mingzhe Li, Yuheng Ji, Chenxi Wang, Zixiang Xu, Zeyu Zhang, Xiaoqing Zhang, Qian Jiang, Zhenhao Chen, et al. Maniplvm-r1: Reinforcement learning for reasoning in embodied manipulation with large vision-language models. *arXiv preprint arXiv:2505.16517*, 2025.

[38] Jiaming Liu, Mengzhen Liu, Zhenyu Wang, Pengju An, Xiaoqi Li, Kaichen Zhou, Senqiao Yang, Renrui Zhang, Yandong Guo, and Shanghang Zhang. Robomamba: Efficient vision-language-action model for robotic reasoning and manipulation. *Advances in Neural Information Processing Systems*, 37:40085–40110, 2024.

[39] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. *Advances in neural information processing systems*, 31, 2018.

[40] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *International Conference on Learning Representations*. International Conference on Learning Representations, ICLR, 2019.

[41] Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14953–14962, 2023.

[42] Tuomo Hiippala, Malihe Alikhani, Jonas Haverinen, Timo Kalliokoski, Evanfiya Logacheva, Serafina Orekhova, Aino Tuomainen, Matthew Stone, and John A. Bateman. Ai2d-rst: A multimodal corpus of 1000 primary school science diagrams. *Language Resources and Evaluation*, 55(3):661–688, 2021.

[43] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *ICLR*, 2024.

[44] Wanqing Cui, Xin Hong, Yanyan Lan, Liang Pang, Jiafeng Guo, and Xueqi Cheng. Visual transformation telling. *arXiv preprint arXiv:2305.01928*, 2023.

[45] Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth a dozen images. In *ECCV*, pages 235–251, 2016.

[46] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. *arXiv preprint arXiv:2310.02255*, 2023.

[47] Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *NeurIPS*, 2022.

Appendix

This appendix provides supplementary materials that complement the main text. It covers related literature, dataset representation and construction, extended statistics, and additional experimental results. The sections are organized as follows:

- Sec. A reviews related work, situating MATHSTICKS within research on advanced visual reasoning, symbolic compositionality, and diagram- or math-based benchmarks.
- Sec. B introduces the visual representation of matchsticks, detailing the segment-level indexing scheme that underlies all symbolic edits.
- Sec. C describes the dataset construction pipeline, including symbolic enumeration, arithmetic filtering, and deterministic visual rendering.
- Sec. D reports detailed dataset statistics with both tabular summaries and distribution visualizations.
- Sec. E presents extended experimental results, including fine-grained analyses and human evaluation.
- Sec. F lists the exact prompts used for evaluation under both text-prompted and pure-visual regimes, enabling full reproducibility.
- Sec. G presents qualitative case studies that illustrate representative model successes and common failure patterns.

A Related Work

Vision–Language Models for Advanced Visual Reasoning. Early work in multimodal learning largely concentrated on perception-oriented objectives such as recognition, captioning, and retrieval, often framed around improving robustness and generalization across diverse inputs [15–19]. With the advent of vision–language models (VLMs) [20–29], the research focus has progressively shifted toward higher-level reasoning beyond static recognition. Recent studies highlight the effectiveness of chain-of-thought prompting and structured exploration strategies for enhancing compositional reasoning [30, 31], while reinforcement-based alignment further improves multi-step logical consistency [32, 33]. Parallel efforts extend these paradigms into embodied manipulation, where VLMs are optimized for long-horizon action planning and reasoning in physical environments [34–38]. Together, these advances suggest a broader trajectory: visual reasoning with large models is evolving from perception-centric tasks toward dynamic, multi-step, and embodiment-aware problem solving.

Visual Symbolic Compositional Reasoning (VSCR). VSCR requires a model to identify symbolic elements in images, plan reachable local edits under explicit constraints, and verify symbolic consistency after editing. This capability extends the scope of traditional compositional visual reasoning, which typically answers queries on a fixed scene without legal editing or post hoc verification. Early benchmarks such as CLEVR established controlled templates for attributes, relations, and counting, and spurred research on modular architectures and program-supervised reasoning [1]. Neuro-symbolic approaches further introduced explicit executors for logical verification [39, 40], while visual programming frameworks leverage large language models to synthesize executable procedures from images, modularizing perception and symbolic manipulation [41]. However, these paradigms usually assume gold programs or fixed structures and rarely require searching for legal, solvable edits in the visual space. In contrast, our setting instantiates VSCR with matchstick equations and enforces stick conservation, restricted move budgets, operator flipping when legal, and arithmetic correctness, including the distinction between unique and multiple solutions.

Visual Transformation and Diagram- and Math-based Reasoning. Benchmarks on transformation and structural reasoning provide useful comparators but address different objectives. CLEVR-Change targets change captioning between image pairs without requiring algebraic verification [7, 42–44]. TRANCE emphasizes transformation-driven reasoning in synthetic worlds but abstracts away from fine-grained symbolic edits and correctness guarantees [6]. VisualTrans studies real-world human–object interactions, focusing on functional consequences of local modifications rather than constrained symbolic correction [7]. Datasets on diagrams and mathematics highlight complementary reasoning skills. AI2D and AI2D_RST examine parsing and relational understanding of instructional

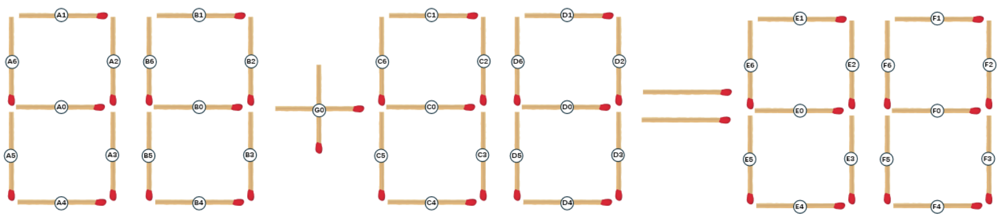


Figure 2: Figure 2: Illustration of the segment-level indexing scheme. Each digit position in the equation (indexed sequentially from left to right) is decomposed into seven labeled segments (0–6).

diagrams [42, 45]. MathVista aggregates diverse visual math problems to test language–math compositionality [46], while ChartQA and ScienceQA extend evaluation to chart-based reasoning and multimodal science questions [5, 47]. Although valuable, these benchmarks primarily measure image-to-text answering. They seldom require a model to enumerate legal edits in the visual space and to verify symbolic correctness after modification.

MATHSTICKS complements these efforts by operationalizing VSCR as an end-to-end pipeline that integrates symbol recognition, constrained transformation planning, and arithmetic verification. It provides unified rendering rules, controllable difficulty slices (digit scale, move complexity, solution multiplicity, operator flipping), and large-scale coverage, enabling fine-grained diagnosis of visual–symbolic reasoning beyond prior QA- or captioning-centric tasks.

B Visual Representation of Matchsticks

To enable reproducible data generation and unambiguous parsing, we introduce a *segment-level indexing scheme* for all matchstick configurations. Each digit is rendered in a seven-segment layout, and every possible matchstick location is assigned a unique identifier, as shown in Fig. 2.

Digit labeling. For each operand and result digit, we adopt the standard seven-segment convention with indices $\{0, 1, \ldots, 6\}$ referring to the horizontal and vertical strokes. As illustrated in Fig. 2, segment indices are prefixed with a letter denoting the digit slot (A for the first digit, B for the second digit, etc.). For example: - `A0` denotes the bottom horizontal segment of digit A, - `A1` the top horizontal segment, - `A2` the top-right vertical segment, - `A3` the bottom-right vertical segment, - `A4` the bottom horizontal, - `A5` the bottom-left vertical, - `A6` the top-left vertical. The same scheme applies for all subsequent digit slots (B0–B6, C0–C6, etc.).

Operator labeling. The operator is placed in a dedicated slot G. We fix the horizontal bar as immutable, and define only one editable segment `G0` for the vertical stroke. When `G0` is present, the operator is "$+$"; when absent, the operator is "$-$". Thus operator changes are realized by adding or removing the single stick at `G0`. The equality sign "$=$" is only a visual separator, not indexed and not involved in edits.

Equation layout. Each equation instance thus corresponds to a set of labeled matchsticks:

$$\{\texttt{A0-A6}\} \cup \{\texttt{B0-B6}\} \cup \{\texttt{C0-C6}\} \cup \{\texttt{D0-D6}\} \cup \{\texttt{E0-E6}\} \cup \{\texttt{F0-F6}\}$$

for digits, combined with operator segments (`G0`) and equality segments. This explicit labeling allows us to formalize puzzle states as discrete vectors, supporting precise move operations such as

`Move(A0, C3)`

which denotes relocating the stick from position `A0` to position `C3`.

Usage in prompt construction. This indexing scheme is also embedded in the evaluation prompt (see Sec. F), where models are required to output moves in the canonical format `Move(<source>, <target>)`. The visual-to-symbolic mapping ensures that predictions are parsable, verifiable, and independent of rendering details.

Algorithm 1: Symbolic Enumeration and Solution Mining

Input: Search ranges for $a, c, e \in \{-1, \ldots, 9\}$ (tens slots), $b, d, f \in \{0, \ldots, 9\}$ (units), and $g \in \{+, -\}$.
Output: A set $\mathcal{D}$ of solvable instances with diagnostic labels.

$\mathcal{D} \leftarrow \varnothing$
foreach a, b, g, c, d, e, f *in the Cartesian product* **do**
  $\mathbf{z} \leftarrow [a, b, g, c, d, e, f]$
  $(g^*, A, B, C) \leftarrow$ `SoloToWhole`$(\mathbf{z})$
  if `IsValidArithmetic`$((g^*, A, B, C))$ **then**
    continue
    `// Original is already valid; skip as source`
  $S_1 \leftarrow$ `EnumerateOneStick`$(\mathbf{z}; \mathcal{T}_1)$
  $S_2 \leftarrow$ `EnumerateTwoSticks`$(\mathbf{z}; \mathcal{T}_2)$
  `// Arithmetic filtering`
  $S_1^{\checkmark} \leftarrow \{\mathbf{z}' \in S_1 :$ `IsValidArithmetic(SoloToWhole(`$\mathbf{z}'$`))`$\}$
  $S_2^{\checkmark} \leftarrow \{\mathbf{z}' \in S_2 :$ `IsValidArithmetic(SoloToWhole(`$\mathbf{z}'$`))`$\}$
  `// Deduplicate 2-stick solutions that also appear in 1-stick`
  $S_2^{\star} \leftarrow S_2^{\checkmark} \setminus S_1^{\checkmark}$
  if $|S_1^{\checkmark}| + |S_2^{\star}| > 0$ **then**
    $\ell \leftarrow$ `AssignLabels`$(\mathbf{z}, S_1^{\checkmark}, S_2^{\star})$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{z}, S_1^{\checkmark}, S_2^{\star}, \ell)\}$
return $\mathcal{D}$

C Dataset Construction Details

To ensure both completeness in the symbolic search space and fidelity in the visual representation, we propose a two-stage construction pipeline. First, *symbolic enumeration* systematically explores all candidate equations and mines valid solutions under one- and two-stick moves. Second, *visual rendering* deterministically assembles equation images from manually designed segment templates. This design guarantees large-scale coverage with precise alignment between symbolic transformations and visual stimuli.

C.1 Symbolic Enumeration

State encoding. Each equation instance is represented by a 7-slot tuple $\mathbf{z} = [a, b, g, c, d, e, f]$, where a, b are the tens/units of the first operand, c, d the tens/units of the second operand, e, f the tens/units of the result, and the operator slot $g \in \{+, -\}$. We allow blanks (coded as -1) for the tens slots a, c, e. A helper map SoloToWhole$(\cdot)$ converts $\mathbf{z}$ into $(g,\ A,\ B,\ C)$ with $A = 10{\cdot}\max(a, 0) + b$, $B = 10{\cdot}\max(c, 0) + d$, $C = 10{\cdot}\max(e, 0) + f$. Arithmetic validity is then checked directly: the equation is valid if $A + B = C$ when $g{=}{+}$, or if $A - B = C$ when $g{=}{-}$.

Move space. Seven-segment digits follow the standard 7-stick layout. For each digit slot (A,B,C,...) and segment index $s \in \{0, \ldots, 6\}$, a labeled stick position (e.g., A0, A1, ..., A6) is defined (see Fig. 2). Two lookup tables encode legal edits: $\mathcal{T}_1$ for all *single-stick* moves and $\mathcal{T}_2$ for all *two-stick* composite edits (including within-digit changes, cross-digit transfers, and operator flips), all under stick conservation.

Operator labeling. The operator occupies a dedicated slot G. Its horizontal bar is fixed, while the vertical stroke is the only editable segment and is indexed as `G0`. When `G0` is present the operator is "+"; when absent it is "−". Thus operator changes are realized by adding or removing the stick at `G0`. The equality symbol "=" is only a visual separator in figures; it is not indexed and does not participate in edits.

Solution mining. For each $\mathbf{z}$ we enumerate: (i) all 1-stick reachable states $S_1(\mathbf{z})$; (ii) all 2-stick reachable states $S_2(\mathbf{z})$. After arithmetic filtering we deduplicate $S_2(\mathbf{z}) \setminus S_1(\mathbf{z})$. Each surviving

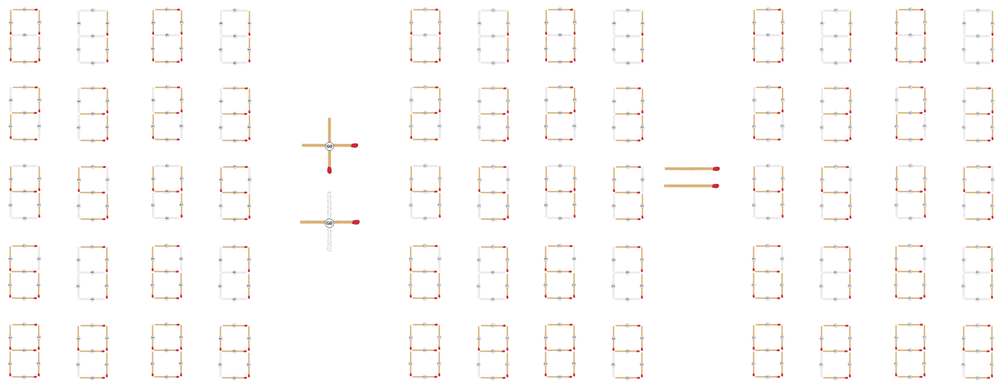


Figure 3: Example of the template library, showing digit slots with indexed segments and the operator slot. Each index corresponds to a movable matchstick.

Algorithm 2: Deterministic Visual Assembly

Input: Symbolic equation $\mathbf{z} = [a, b, g, c, d, e, f]$; template library $\mathcal{L}$ with per-slot segment-indexed assets.

Output: Rendered image $\mathsf{Img}(\mathbf{z})$.

$\mathcal{G} \leftarrow []$ // glyph list

foreach *slot* $s \in \{A, B, G, C, D, E, F\}$ **do**

  $v \leftarrow$ value in $\mathbf{z}$ at slot s

  $\mathcal{G}$.append(`FetchGlyph`($\mathcal{L}, s, v$))

$\mathsf{Row} \leftarrow$ `Concat`($\mathcal{G}$) // fixed kerning and baseline

$\mathsf{Img}(\mathbf{z}) \leftarrow$ `Layout`(Row)

return $\mathsf{Img}(\mathbf{z})$

transformation is labeled with (digit scale: number of two-digit operands/results), (move complexity: 1- vs. 2-stick), (solution multiplicity), and (operator flipping: whether $+ \leftrightarrow -$).

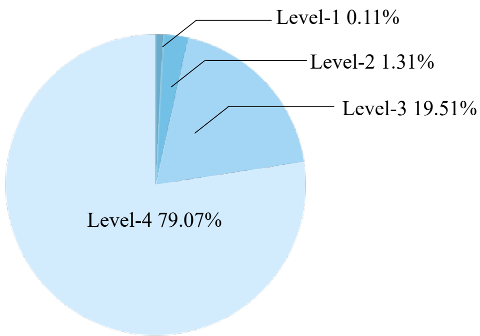
The overall enumeration and filtering procedure is summarized in Alg. 1.

C.2 Visual Rendering

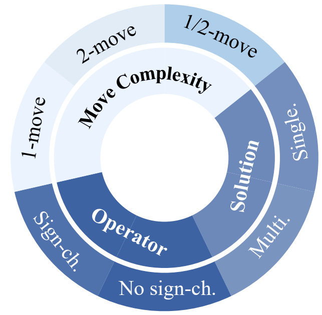
Template library. To standardize the visual construction, we manually created seven-segment digits (0–9) and the operator slot using a vector graphics editor. For each digit position (A, B, C, ...), we assigned unique indices to its seven possible segments, denoted as `A0` ... `A6`, `B0` ... `B6`, etc., ensuring one-to-one correspondence between indices and visible sticks. Each segment was drawn as an independent graphical object, and the operator slot was treated analogously (e.g., index `G0` for the horizontal bar). All assets were exported to PNG/SVG with fixed canvas size, baseline alignment, and uniform spacing. An example of the template library is illustrated in Fig. 3.

Deterministic assembly. Given a symbolic equation $\mathbf{z} = [a, b, g, c, d, e, f]$, rendering proceeds by slot-wise retrieval and horizontal concatenation of the corresponding templates for slots $\{A, B, G, C, D, E, F\}$. Because indices are consistent across slots, a symbolic edit such as MOVE(`A0`, `C3`) corresponds to a unique visible stick relocation. The full procedure is summarized in Alg. 2.

Reproducibility. All coordinates (origins, slot offsets, segment bounding boxes) are stored with the templates. Thus any symbolic transformation applied to solvable instances $\mathcal{D}$ (from the construction step in Sec. 1) can be re-rendered without manual intervention, ensuring consistent alignment between symbolic and visual forms.



(a) Distribution across difficulty levels.



(b) Distribution across task dimensions.

Figure 4: **Dataset distribution.** (a) Proportions across difficulty levels. (b) Decomposition by move complexity, solution multiplicity, and operator flipping.

Table 3: **Detailed dataset statistics.** Counts and percentages by difficulty level, further broken down by move complexity, solution multiplicity, and operator flipping.

Level	Move Complexity			Solution Multiplicity		Operator	
	1-move	2-move	1/2-move	Unique	Multiple	Flip	No flip
L1 (1,505)	202 (13.42%)	880 (58.47%)	423 (28.11%)	548 (36.41%)	957 (63.59%)	819 (54.42%)	686 (45.58%)
L2 (18,466)	1,875 (10.15%)	14,340 (77.66%)	2,251 (12.19%)	11,692 (63.32%)	6,774 (36.68%)	6,743 (36.52%)	11,723 (63.48%)
L3 (275,406)	15,348 (5.57%)	219,715 (79.78%)	40,343 (14.65%)	127,208 (46.19%)	148,198 (53.81%)	105,185 (38.19%)	170,221 (61.81%)
L4 (1,116,011)	41,505 (3.72%)	922,571 (82.67%)	151,935 (13.61%)	469,204 (42.04%)	646,807 (57.96%)	405,810 (36.36%)	710,201 (63.64%)
Total (1,411,388)	58,930 (4.18%)	1,157,506 (82.01%)	194,952 (13.81%)	608,652 (43.12%)	802,736 (56.88%)	518,557 (36.74%)	892,831 (63.26%)

D Detailed Dataset Statistics

This section provides a comprehensive breakdown of the MathSticks dataset, complementing the summary in the main text. In total, the benchmark contains 1,411,388 solvable instances. We analyze their distribution along three orthogonal axes: difficulty levels, move complexity, and solution multiplicity, as well as operator-flip requirements. Fig. 4 visualizes these dimensions, and Tab. 3 reports detailed counts and percentages.

Difficulty levels. As shown in Fig. 4(a), Level 4 dominates the dataset (79.07%), while Levels 1–3 contribute 0.11%, 1.31%, and 19.51%, respectively. This skew is not an artifact of our sampling methodology but rather an intrinsic characteristic of the problem's combinatorial nature. The number of solvable equations scales combinatorially with the introduction of multi-digit operands, as they significantly enlarge the state space of valid transformations under stick conservation rules. Consequently, the full 1.4M dataset provides an unbiased representation of the natural frequency distribution of puzzle complexities inherent to this domain.

Move complexity. Most instances require two-stick transformations (82.01%), with one-stick solutions accounting for only 4.18%. The remaining 13.81% can be solved by either a one-stick or two-stick move, illustrating the presence of multiple valid correction paths. This composition highlights the dataset's emphasis on composite reasoning over local single-edit corrections.

Solution multiplicity. A substantial fraction of puzzles admit multiple valid edits. Across the dataset, 56.88% of instances fall into this category, while 43.12% have a unique correction. Multi-solution cases are particularly challenging for autoregressive models, which must converge on one valid output despite the ambiguity.

Operator flipping. Tasks involving operator changes ($+ \leftrightarrow -$) form a critical subspace, requiring models to edit abstract symbolic elements in addition to digit morphology. This dimension further stresses the need for integrating symbolic reasoning beyond perceptual transformation.

Table 4: **Results on the MathSticks Benchmark (w/ text prompt).** Accuracy (%) is reported across categories reflecting move complexity, solution multiplicity, and operator change. The best performance is in **bold**, the second best is <u>underlined</u>.

Model	Move Complexity			Solution Multiplicity		Operator Change	
	1-move	2-move	1/2-move	Single-sol.	Multi-sol.	Sign-changed	No sign-ch.
Closed Models							
o3-250416	**82.50**	**52.88**	**89.18**	**47.79**	**71.69**	**51.45**	**65.19**
Gemini-2.5-Pro-250506	<u>51.67</u>	<u>38.43</u>	<u>76.37</u>	<u>28.93</u>	<u>59.42</u>	<u>38.91</u>	<u>49.09</u>
Gemini-2.5-Flash-250520	43.89	18.20	62.66	15.29	36.96	23.94	29.20
GPT-o4-mini-250416	59.17	14.71	66.19	15.60	33.05	21.15	26.95
Seed-1.6-Thinking-250615	31.39	15.17	41.72	10.89	30.06	16.83	24.80
Seed-1.6-250615	7.78	1.36	9.40	1.96	4.21	4.76	1.92
Claude Sonnet 4	7.78	0.78	5.63	1.21	2.81	2.85	1.57
GPT-4o-241120	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Open-Source Models							
Qwen2.5-VL-7B-Instruct	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Qwen2.5-VL-32B-Instruct	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Qwen2.5-VL-72B-Instruct	0.00	0.00	0.00	0.00	0.00	0.00	0.00
InternVL3-8B	0.00	0.00	0.00	0.00	0.00	0.00	0.00
InternVL3-38B	0.00	0.00	0.00	0.00	0.00	0.00	0.00
InternVL3-78B	0.00	0.00	0.00	0.00	0.00	0.00	0.00

E More Experiments

E.1 Evaluation Protocol

All models, including both closed-source and open-source families, were evaluated via their respective public APIs to ensure a consistent testing environment. We utilized the default API settings for all models in our experiments. Our protocol did not set an explicit `max_token` limit, allowing models to generate their full reasoning without premature termination.

E.2 Fine-Grained Analysis

To further dissect model behavior, we evaluate performance across fine-grained categories that capture different aspects of reasoning complexity. Specifically, we consider three orthogonal dimensions: *move complexity* (one-stick vs. two-stick vs. mixed), *solution multiplicity* (single-solution vs. multiple-solution instances), and *operator flipping* (addition vs. subtraction). Tab. 4 and Tab. 5 summarize results under the text-prompted and pure-visual regimes, respectively.

Move complexity. Two-stick puzzles are consistently more difficult than one-stick puzzles across all models. The performance gap is particularly striking in weaker closed-source models, where accuracy often drops below 20%. This reflects the challenge of planning and executing composite edits, which requires maintaining consistency across multiple segments. In contrast, one-stick puzzles involve simpler local edits and thus remain more tractable. Mixed cases (one/two-move) typically yield higher scores, since many can be solved with an easier one-stick transformation even when a two-stick option exists.

Solution multiplicity. Problems admitting multiple valid corrections prove substantially harder. Even strong models such as GPT-o3 and Gemini 2.5 Pro exhibit drops of 20–30 points compared with single-solution cases. This indicates that models are highly sensitive to ambiguity in the solution space: when several structurally distinct transformations are possible, they often fail to converge on a correct candidate. By contrast, single-solution instances provide a unique correction target, which better aligns with the deterministic nature of autoregressive decoding.

Operator flipping. Tasks requiring operator changes ($+ \leftrightarrow -$) constitute a major bottleneck. Performance is consistently lower than in no-flip cases, suggesting that models are biased toward digit-level manipulations rather than considering operator edits. This weakness is particularly pronounced in weaker closed models and universal across all open-source baselines, which almost entirely fail on operator-flip puzzles. These results highlight the difficulty of extending generalization from digit morphology to abstract symbolic transformations.

Table 5: **Results on the MathSticks Benchmark (w/o text prompt).** Accuracy (%) is reported across categories reflecting move complexity, solution multiplicity, and operator change. The best performance is in **bold**, the second best is underlined.

Model	Move Complexity			Solution Multiplicity		Operator Change	
	1-move	2-move	1/2-move	Single-sol.	Multi-sol.	Sign-changed	No sign-ch.
Closed Models							
o3-250416	**70.00**	**32.12**	**56.70**	**31.52**	**45.63**	**34.20**	**41.22**
Gemini-2.5-Pro-250506	25.83	18.22	36.63	14.76	29.72	19.70	25.05
Gemini-2.5-Flash-250520	18.33	5.92	15.33	4.98	11.89	8.49	8.34
GPT-o4-mini-250416	10.83	7.23	22.16	5.38	13.92	10.54	9.88
Seed-1.6-Thinking-250615	0.00	0.68	1.85	0.00	1.55	1.61	0.58
Seed-1.6-250615	0.00	0.39	0.93	0.00	0.75	0.44	0.58
Claude Sonnet 4	0.00	0.00	0.00	0.00	0.00	0.00	0.00
GPT-4o-241120	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Open-Source Models							
Qwen2.5-VL-7B-Instruct	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Qwen2.5-VL-32B-Instruct	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Qwen2.5-VL-72B-Instruct	0.00	0.00	0.00	0.00	0.00	0.00	0.00
InternVL3-8B	0.00	0.00	0.00	0.00	0.00	0.00	0.00
InternVL3-38B	0.00	0.00	0.00	0.00	0.00	0.00	0.00
InternVL3-78B	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 6: **Human performance on the MathSticks benchmark (w/o text prompt).** Accuracy (%) is reported across difficulty levels together with the average solving time per participant. Since human recognition of digits/operators is error-free, the same accuracies apply under the text-prompted regime.

Participant	Accuracy (%)					Avg. Time
	L1	L2	L3	L4	Avg.	
P1	95.0	100.0	95.0	100.0	97.5	1m14s
P2	100.0	95.0	72.5	73.7	85.3	1m17s
P3	90.0	100.0	90.0	89.5	92.4	1m06s
Mean	95.0	98.3	85.8	87.7	91.7	—
Std.	5.0	2.9	11.8	13.3	6.1	—

Cross-regime comparison. The shift from text-prompted to pure-visual inputs introduces systematic degradation. For example, Gemini 2.5 Pro drops from 51.7% on one-move text-prompted puzzles to 25.8% in the pure-visual setting. This gap underscores OCR and structural parsing as an additional error source, independent of symbolic reasoning itself. GPT-o3, while still affected, demonstrates relatively smaller gaps, suggesting stronger robustness to visual noise and layout variability compared with other models.

Open-source models. All tested open-source models (Qwen2.5-VL and InternVL3 families) fail almost entirely across categories, with accuracies close to zero. This finding emphasizes the current limitations of open-source VLMs in handling structured visual-symbolic reasoning tasks. Notably, increasing model scale does not alleviate the problem: even the largest variants with 70B+ parameters remain at chance-level performance. This suggests that the failure stems not from insufficient capacity, but from the lack of targeted supervision and inductive biases for visual–symbolic reasoning. It further points to the need for specialized training data and architectural innovations capable of bridging continuous perception with discrete symbolic manipulation.

E.3 Human Evaluation

To complement model-based evaluation, we conducted a small-scale human study on the MathSticks benchmark. Three adult participants (all male, aged 24–30, with university-level education backgrounds) independently solved the full evaluation set under the **pure-visual regime**, where only the rendered matchstick equations were provided. Because the digits and operators are visually

Prompt with text input

Task:
You are given an incorrectly displayed equation "{equation}" constructed from matchsticks in a seven-segment format. Each segment (with matchstick or without matchstick) is labeled with a **unique identifier** (e.g., A0, C4). Your goal is to modify the equation by moving **one or two matchsticks** to make it **mathematically correct.**

Constraints:
- Only reposition existing matchsticks (no addition/removal).
- Only one or two matchsticks can be moved, and each matchstick can only be moved once.
- The final equation must be mathematically valid.
- Preserve digit legibility (no broken/unrecognizable characters).

Output Requirements:
1. Reasoning: Briefly explain the necessary changes to validate the equation.
2. Final Solution Format: Provide the answer strictly in the format:
  - For single move: `boxed{{Move(<original_label>, <target_label>)}}`, for example: `boxed{{Move(B2, D5)}}`
  - For two moves: `boxed{{Move(<original_label1>, <target_label1>), Move(<original_label2>, <target_label2>)}}`, for example: `boxed{{Move(A0, C3), Move(E1, F4)}}`

Note:
Oberserve the given image carefully to identify the matchstick positions and their labels.
Gray dashed segments indicate no matchstick, while solid segments indicate a matchstick is present.
Ensure the final **answer adheres precisely to the `boxed{{}}` format** for automated parsing.

Figure 5: **Prompt with text input.** The symbolic equation string is provided together with the matchstick rendering.

unambiguous, participants transcribed the equations without error; hence, the same results apply to the text-prompted regime as well.

Tab. 6 summarizes participant-level accuracies across difficulty levels together with average solving times. On average, humans achieved accuracies above 90% while spending roughly one minute per problem. These results confirm that the benchmark is reliably solvable by humans, yet remains highly challenging for current vision–language models.

F Prompts

To ensure reproducibility, we provide the exact prompts used in our evaluation. Two regimes are supported: (*i*) **text-prompted**, where the symbolic equation string is explicitly given; and (*ii*) **pure-visual**, where only the rendered matchstick image is shown, requiring OCR and structural parsing.

G Case Studies

To qualitatively illustrate model behavior, we sampled two representative puzzles from each difficulty level (L1–L4), under both input regimes (text-prompted and pure-visual), resulting in 16 cases in total. For each case, we present the input matchstick puzzle (i.e., the incorrect equation to be solved) and compare predictions from multiple representative models. Correct solutions are not explicitly annotated in the figure but can be verified against the solvability constraints of the benchmark. This analysis highlights common failure modes, including: incorrect operator flipping, partial edits that leave digits illegible, and failure to converge when multiple solutions exist. Fig. 7–Fig. 22 present the visualizations.

Prompt without text input

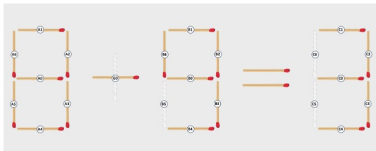
Task:
You are given an incorrectly displayed equation constructed from matchsticks in a seven-segment format , as shown in the given image. Each segment (with matchstick or without matchstick) is labeled with a **unique identifier** (e.g., A0, C4). Your goal is to modify the equation by moving **one or two matchsticks** to make it **mathematically correct.**

Constraints:
- Only reposition existing matchsticks (no addition/removal).
- Only one or two matchsticks can be moved, and each matchstick can only be moved once.
- The final equation must be mathematically valid.
- Preserve digit legibility (no broken/unrecognizable characters).

Output Requirements:
1. Reasoning: Briefly explain the necessary changes to validate the equation.
2. Final Solution Format: Provide the answer strictly in the format:
   - For single move: `boxed{{Move(<original_label>, <target_label>)}}`, for example: `boxed{{Move(B2, D5)}}`
   - For two moves: `boxed{{Move(<original_label1>, <target_label1>), Move(<original_label2>, <target_label2>)}}`, for example: `boxed{{Move(A0, C3), Move(E1, F4)}}`

Note:
Oberserve the given image carefully to identify the matchstick positions and their labels.
Gray dashed segments indicate no matchstick, while solid segments indicate a matchstick is present.
Ensure the final **answer adheres precisely to the `boxed{{}}` format** for automated parsing.

Figure 6: **Prompt without text input.** Only the matchstick rendering is provided; models must recognize symbols before reasoning.



Gemini-2.5-Pro

With Text Input：

Reasoning:
The initial equation displayed by the matchsticks is `8 - 9 = 3`, which is mathematically incorrect. The goal is to make the equation valid by moving two matchsticks.
The solution is to transform the equation into `9 - 9 = 0`.
1. Transform '8' to '9': The first digit '8' (composed of all seven segments A0 -A6) can be changed into a '9' by removing the top-left matchstick, which is labeled `A6`.
2. Transform '3' to '0': The third digit '3' (composed of segments C0, C1, C2, C3, C4) can be changed into a '0' (composed of segments C1, C2, C3, C4, C5, C6). This requires removing the middle stick `C0` and adding sticks at the top -left (`C6`) and bottom-left (`C5`) positions … After these two moves, the first digit is a '9', the second digit remains a '9', the operator is unchanged, and the third digit becomes a '0'. The resulting equation is `9 - 9 = 0`, which is correct.
Final Solution Format:
\\boxed{Move(A6, C6), Move(C0, C5)}

o3

With Text Input：

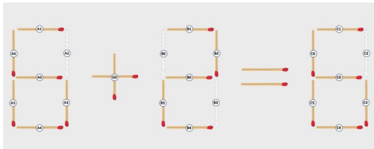
Reasoning:
• Shift the upper-right match of the "9" (segment B2) to its vacant lower -left position (B5). This removes the upper-right stroke and adds the lower-left stroke, turning 9 into 6.
• Shift the lower-right match of the "3" (segment C3) to its vacant lower -left position (C5). This removes the lower-right stroke and adds the lower-left stroke, turning 3 into 2.
After these two moves the equation reads 8 − 6 = 2, which is mathematically correct.
\\boxed{Move(B2, B5), Move(C3, C5)}

Doubao-Seed-1.6

With Text Input：

To solve the problem of transforming the equation "8 -9=3" into a valid one by moving 1 or 2 matchsticks, we analyze the structure of the digits and operator. The key insight is to adjust the digits 8 and 9 to form 9 and 6, respectively, resultin g in the equation "9-6=3".
Reasoning:
-8 to 9: The digit 8 uses all 7 segments. To convert it to 9, we remove the upper -left segment (labeled A5, which is the "e" segment in seven-segment notation). This leaves 8 with 6 segments, forming 9.
- 9 to 6: The digit 9 has 6 segments (missing the upper -left "e" segment). To convert it to 6, we need to add the "e" segment (by moving the segment removed from 8) and remove the upper -right "b" segment (labeled B2). The removed "b" segment from 9 is moved to the lower-left "f" segment of the digit 3 (labeled C5), but since 3 is already correct, the critical moves are transferring the "e" segment from 8 to 9 and removing the "b" segment from 9.
Solution:
Move two matchsticks:
1. Move the upper-left segment (A5) from 8 to the upper-left segment (B5) of 9, converting 8 to 9 and starting the conversion of 9 to 6.
2. Move the upper-right segment (B2) from 9 to the lower-left segment (C5) of 3 (though 3 remains correct, this adjustment completes the conversion of 9 to 6 by removing its excess segment).
Final Answer:
\\boxed{Move(A5, B5), Move(B2, C5)}

Figure 7: Case study (Level 1, text-prompted, Case A).



Gemini-2.5-Pro

With Text Input：

Reasoning:
1. Analyze the initial equation: The image displays the incorrect equation `6 + 2 = 6`. The correct sum of `6 + 2` is `8`.
2. Identify the goal:The task is to make the equation mathematically correct by moving one or two matchsticks.
3. Explore potential solutions:
One possible approach is to keep the first number and the operator the same and change the second number and the result. Let's aim for the equation `6 + 3 = 9`.
4. Execute the moves:
First Move (Changing 2 to 3): To change the digit '2' to a '3', we need to move the bottom -left vertical matchstick (at position B5) to the bottom-right vertical position (B3). The digit '2' is formed by sticks {B1, B2, B0, B5, B4}, and '3' is formed by {B1, B2, B0, B3, B4}. So, the first move is `Move(B5, B3)`….
Final Solution Format:
boxed{Move(B5, B3), Move(C5, C2)}

GPT-4o

With Text Input：

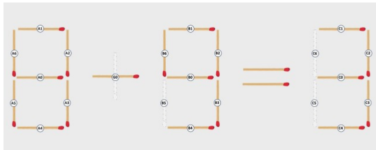
Reasoning:
The equation "6+2=6" is incorrect. To make it mathematically valid, we can adjust the matchstick arrangement to transform one or more digits into new values, ensuring the equation is correct.
By analyzing the problem, one possible solution involves:
1. Transforming the "6" on the left -hand side into an "8".
2. Leaving the "+2" unchanged …
Solution:
1. Move matchstick A2 (from the top -right of the left 6) to position A0 (completing the middle segment of left 8).
2. Move matchstick C2 (from the top -right of the right 6) to position C3 (completing the middle segment of right 8).
3. Final Solution Format:
boxed{Move(A2, A0), Move(C2, C3)}

Claude-Sonnet-4

With Text Input：

I need to analyze the current equation "6+2=6" and find a way to make it mathematically correct by moving one or two matchsticks.
Let me first identify what digits are currently displayed:
- First digit: 6 (segments present
- Operation: + (plus sign)
- Second digit: 2 (segments present)
- Equals sign: = (present)
- Result digit: 6 (segments present)
The equation 6+2=6 is incorrect since 6+2=8.
Let me consider possible solutions by moving matchsticks:
Option 1: Change the result from 6 to 8
To change 6 to 8 … This requires moving one matchstick to add the middle horizontal segment to the rightmost digit, converting the 6 into an 8.
boxed{Move(B0, C0)}

Figure 8: Case study (Level 1, text-prompted, Case B).



Gemini-2.5-Pro

Without Text Input：

Reasoning:
1. Analyze the initial equation: The image displays the equation `8 + 6 = 5` constructed from matchsticks.
The first number is an 8, formed by all seven segments (A0, A1, A2, A3, A4, A5, A6)…
The equation `8 + 6 = 5` is mathematically incorrect.
2. Identify the required change: The goal is to form a valid equation by moving one or two matchsticks. A simple solution can be found by moving a single matchstick.
3. Formulate the solution:
Take the middle horizontal matchstick from the `8`, which is labeled A0. Removing this matchstick changes the `8` into a `0`…
4. Verify the final equation: After moving the matchstick from A0 to C6, the new equation becomes `0 + 6 = 6`. This is a mathematically correct statement. This solution satisfies the condition of moving only one matchstick.

Final Solution Format:
boxed{{Move(A0, C6)}}

o3

Without Text Input：

Reasoning:
• Take away the upper-right matchstick of the first digit (segment A2).
– Without that segment the "8" becomes a "6".
– Place this same stick on the empty vertical position of the operator, turning the " –" into a "+".
• Take away the upper-left matchstick of the middle digit (segment B6).
– Removing it changes the "9" into a "3".
– Put this stick on the empty upper-left position of the last digit (segment C6); the "3" then becomes a "9".
After these two moves the equation reads 6 + 3 = 9, which is mathematically correct.

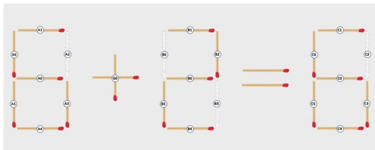
Final Solution:
boxed{Move(A2, G1), Move(B6, C6)}

Doubao-Seed-1.6

Without Text Input：

To solve the problem, we need to correct the equation by moving one or two matchsticks. The original equation, constructed with matchsticks in a seven-segment format, is initially invalid. By analyzing the segments, we identify that the key is to adjust the digits and operator to form a valid equation.
Key Observations:
- The first digit (A) is an 8 (all 7 segments present: A0-A6).
- The second digit (B) is a 1 (segments B2 and B3 present, vertical right segments).
- The result digit (C) is initially incomplete (missing middle and left vertical segments).
- The operator is likely '+' (requiring a vertical segment) but needs to be adjusted to ' -' or the digits modified to form a correct equation…
Final Solution:
The critical move is adjusting the operator by relocating a vertical segment from A to form a valid ' -', resulting in the equation 8 - 1 = 7.
boxed{{Move(A2, C5)}}

Figure 9: Case study (Level 1, pure-visual, Case A).



Gemini-2.5-Pro

Without Text Input：

1. Reasoning:
The initial equation displayed is `6 + 2 = 6`, which is mathematically incorrect. To correct this, a single matchstick needs to be moved.
The solution is to change the equation to `8 - 2 = 6`. This can be achieved with one move:
- Take the vertical matchstick from the plus sign (`+`). The label for the plus sign is `G0`, so we will refer to this stick wi th this label.
- Moving the vertical stick from the `+` sign changes it to a minus sign (` -`)…
Final Solution Format:
\\boxed{{Move(G0, A2)}}

GPT-4o

Without Text Input：

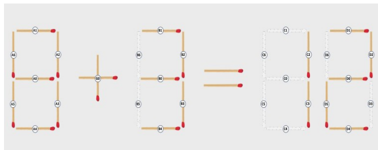
Reasoning:
The given matchstick equation is: `8 + 8 = 8`
This equation is mathematically incorrect.
By observing the matchstick arrangement, we can modify one or two matchsticks to produce a valid equation. The goal is to preserve the seven-segment representation of all digits after moving the matchsticks.
- If we move two matchsticks, we can transform the equation to:
`8 + 3 = 11`, which is mathematically correct . Steps:
1. To change the leftmost `8` to a `3`, remove matchsticks `A6` and `A5`…
Final Solution Format:
\\boxed{{Move(A5, G0), Move(C6, G1)}}

Claude-Sonnet-4

Without Text Input：

I need to analyze the current equation shown in the image and determine how to make it mathematically correct by moving one or two matchsticks.
Looking at the image, I can see a seven-segment display equation. Let me identify what's currently shown:
The equation appears to be: 8 + 8 = 0
This is clearly incorrect mathematically. Let me analyze the segments:
Left side: First digit appears to be 8 (all segments filled)
Middle: + sign \nSecond digit: appears to be 8 (all segments filled)
Equals sign: =
Right side: appears to be 0
To make this equation correct, I need to change one of the numbers. Some possibilities:
- Change 8 + 8 = 0 to 8 - 8 = 0 (change + to -)
- Change one of the 8s to make a valid equation…
boxed{Move(A0, C1)}

Figure 10: Case study (Level 1, pure-visual, Case B).



Gemini-2.5-Flash

With Text Input：

The given equation is `8 + 3 = 12`.
Mathematically, `8 + 3 = 11`, so the equation `8 + 3 = 12` is incorrect.
We need to make it correct by moving one or two matchsticks, ensuring all digits remain legible.Let's try to change the equation to `6 + 6 = 12`.
Current state:
8 (A): Segments A1, A2, A3, A4, A5, A6, A0 are present (all segments).
3 (B): Segments B1, B2, B0, B3, B4 are present. Segments B5, B6 are dashed (missing).
12 (C, D):
1 (C):Segments C2, C3 are present. Segments C1, C4, C5, C6, C0 are dashed…
boxed{Move(A2, B5), Move(B2, B6)}

o4-mini

With Text Input：

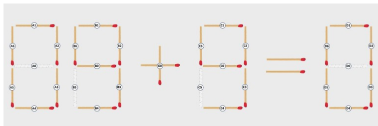
Here's a one-move fix that makes the arithmetic true:
1. Take the lower–left segment of the "8" (segment A5) and place it on the "1" in the result (segment C6).
- Removing A5 turns the left "8" into a "9."
- Adding it at C6 turns the right "1" into a "2," so the result stays "12."
After the move the display reads
9 + 3 = 12
which is correct.
Answer:
boxed{Move(A5, C6)}

Doubao-Seed-1.6-Thinking

With Text Input：

To solve the problem, we need to correct the incorrectly displayed matchstick equation by moving one or two matchsticks. The original equation is assumed to be of the form "X + Y = Z" where the digits are represented using seven -segment displays.
Reasoning
1. Original Equation Analysis: The leftmost digit (labeled A) is an 8 (all segments present: A0 -A6). The middle digit (labeled B) is a 3 (segments B1, B2, B3, B4, B0 present; B5, B6 absent). The rightmost digit (labeled C) is a 5 (segments C1, C3, C4, C6, C0 present; C2, C5 absent). This forms the incorrect equation "8 + 3 = 5".
2. Correcting the Equation: To make the equation valid, we need to swap the values of the leftmost and rightmost digits. This requires moving two matchsticks from the leftmost digit (8) to the rightmost digit (5): - Move the upper right vertical segment (A2) of the 8 to the upper right vertical segment (C2) of the 5…
Result: The corrected equation is "5 + 3 = 8", which is mathematically valid.
Final Solution:
\\boxed{Move(C6, C2), Move(D2, D6)}

Figure 11: Case study (Level 2, text-prompted, Case A).



Gemini-2.5-Pro

With Text Input：
Reasoning:
The original equation is `09 + 9 = 0`, which is mathematically incorrect as `99` is not equal to `0`. To correct this equation by moving two matchsticks, we can transform it into `00 + 9 = 9`.
1. Change the first '9' to a '0': The first '9' is formed by matchsticks at positions B0, B1, B2, B3, B4, and B6. A '0' is formed by matchsticks at B1, B2, B3, B4, B5, and B6. This can be achieved by moving the matchstick from the middle horizontal position (B0) to the bottom-left vertical position (B5).
2. Change the final '0' to a '9': The final '0' is formed by matchsticks at positions D1, D2, D3, D4, D5, and D6. A '9' is formed by matchsticks at D0, D1, D2, D3, D4, and D6. This can be achieved by moving the matchstick from the bottom-left vertical position (D5) to the middle horizontal position (D0).
These two moves result in the arithmetically correct equation `00 + 9 = 9` (which is equivalent to `0 + 9 = 9`).
Final Solution Format:
\\boxed{Move(B0, B5), Move(D5, D0)}

o3

With Text Input：

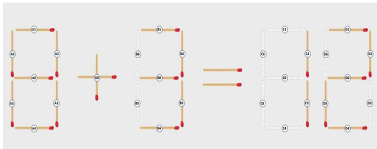
Reasoning:
Each “9” is missing its bottom-left segment (5) and therefore can be turned into a “0” by relocating its middle segment (0) down to that vacant bottom-left position (5).
• Move the middle bar of the first “9” (segment B0) to its empty bottom-left position B5 → the first number “09” becomes “00”…
Solution:
\\boxed{Move(B0, B5), Move(C0, C5)}

Doubao-Seed-1.6

With Text Input：

To solve the problem of correcting the equation "09+9=0" by moving one or two matchsticks, we analyze the structure of the seven-segment display and operators. The goal is to form a valid equation by repositioning matchsticks (solid segments) to dashed segments.
Key Insight:
The original equation "09+9=0" can be transformed into "9-9=0" by modifying two elements:
1. Change the first digit "0" to "9" by adding the middle segment (which 0 lacks but 9 requires).
2. Change the "+" operator to "-" by removing its vertical segment (the "+" has two segments: horizontal and vertical; removing the vertical leaves "-").
Analysis:
- First Digit (0 to 9): A "0" in seven-segment display lacks the middle segment (dashed). To become "9", this middle segment must be filled. Assume the middle segment of the first 0 is labeled A6 (dashed, no matchstick).
- Operator (+ to -): The "+" operator has a vertical segment that is present (solid). Removing this vertical segment (by moving it) converts "+" to "-". Assume this vertical segment is labeled C0 (solid, has matchstick).
Solution:
Move the vertical segment of the "+" (C0) to the middle segment of the first "0" (A6). This converts the first "0" to "9" and the "+" to "-", resulting in the valid equation "9-9=0".
Final Answer:
\\boxed{Move(C0, A6)}

Figure 12: Case study (Level 2, text-prompted, Case B).



Gemini-2.5-Flash

Without Text Input：

The original equation displayed is: 9 + 3 = ?4
Let's analyze the digits based on the provided matchstick configurations:
First Digit (A): Segments A1, A2, A0, A3, A4, A5 are present. Segment A6 is missing (dashed). This configuration represents the digit '9'.
Operator (G): G0 represents the '+' sign.
Second Digit (B): Segments B1, B2, B0, B3, B4 are present. Segments B5, B6 are missing (dashed). This configuration represents the digit '3'.
Equals Sign:Correctly displayed…
This solution uses exactly two matchstick moves.
The final answer is $\\boxed{{Move(A0, A6), Move(C0, D3)}}$

o4-mini

Without Text Input：

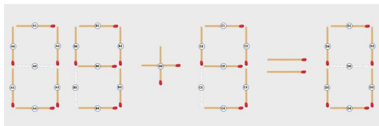
Reasoning:
By removing the middle-bar of the “3” (segment B0) and re-using it as the upper-left segment of the right –hand digit (segment D6), the left-side “83” becomes “87” and the right –side “1 ?” becomes “10” – yielding the true statement 87 = 10.
Final answer:
\\(\\boxed{Move(B0, D6)}\\)

Doubao-Seed-1.6-Thinking

Without Text Input：

To solve the problem, we need to correct the incorrectly displayed matchstick equation by moving one or two matchsticks. The original equation is assumed to be of the form "X + Y = Z" where the digits are represented using seven -segment displays.
Reasoning
1. Original Equation Analysis: The leftmost digit (labeled A) is an 8 (all segments present: A0 -A6). The middle digit (labeled B) is a 3 (segments B1, B2, B3, B4, B0 present; B5, B6 absent). The rightmost digit (labeled C) is a 5 (segments C1, C3, C4, C6, C0 present; C2, C5 absent). This forms the incorrect equation " 8 + 3 = 5".
2. Correcting the Equation: To make the equation valid, we need to swap the values of the leftmost and rightmost digits. This requires moving two matchsticks from the leftmost digit (8) to the rightmost digit (5):
- Move the upper right vertical segment (A2) of the 8 to the upper right vertical segment (C2) of the 5. This changes the 8 to a 5 (losing the upper right segment) and the 5 to an 8 (gaining the upper right segment).
- Move the lower left vertical segment (A5) of the 8 to the lower left vertical segment (C5) of the 5. This completes the transformation of the 8 to a 5 (losing the lower left segment) and the 5 to an 8 (gaining the lower left segment). \n\n3. Result: The corrected equation is "5 + 3 = 8", which is mathematically valid.
Final Solution Format
\\boxed{{Move(A2, C2), Move(A5, C5)}}

Figure 13: Case study (Level 2, pure-visual, Case A).



Gemini-2.5-Pro

Without Text Input：

1. Reasoning:
The initial equation displayed is ` 06 + 6 = 0`. Interpreting the two-digit number `06` as `6`, the equation is `6 + 6 = 0`, which is mathematically incorrect. To solve the puzzle, we can move two matchsticks to form a valid equation.
1. The first move is to take the middle horizontal matchstick from the second '6' (labeled C0) and place it in the empty lower-left position of the same digit (labeled C5). This transforms the digit '6' into a '0'. The equation now reads `06 + 0 = 0`.
2. The second move is to take the upper-right vertical matchstick from the '0' on the right side of the equation (labeled D2) and place it in the empty middle horizontal position of that same digit (labeled D0). This transforms the '0' into a '6'.
The resulting equation is `06 + 0 = 6` (which is `6 + 0 = 6`), a mathematically correct statement.
2. Final Solution Format:
\\boxed{{Move(C0, C5), Move(D2, D0)}}

o3

Without Text Input：

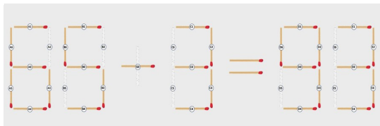
Reasoning:
At the start the equation shows 09 + 9 = 0 (which is wrong).
• If we turn the third digit 9 into 0, and at the same time turn the result 0 into 9, we will have 09 + 0 = 9, i.e. 9 + 0 = 9 , which is correct.
• A 9 becomes 0 by moving its middle segment to the lower -left position.
• A 0 becomes 9 by moving its lower -left segment to the middle position.
Doing these two moves fixes the arithmetic without affecting any other part of the display.
Final Solution Format:
\\boxed{Move(C0, C5), Move(D5, D0)}

Doubao-Seed-1.6

Without Text Input：

To solve the problem, we first identify the original equation using the seven -segment matchstick display. The groups A, B, C, D correspond to digits or operators, with dashed segments indicating missing matchsticks and solid segments indicating present matchsticks.
Key Observations:
- Group A: Segments A1, A2, A3, A4, A5, A6 are solid (middle segment A0 is dashed) → forms the digit `0`.
- Group B: All segments (B0-B6) are solid → forms the operator `+` (cross shape with horizontal and vertical segments).
- Group C: All segments (C0-C6) are solid → forms the digit `8`.
- Group D: Segments D1, D2, D3, D4, D5, D6 are solid (middle segment D0 is dashed) → forms the digit `0`.
- The original equation is thus ` 0 + 8 = 0`, which is mathematically incorrect.
Solution:
To correct the equation, we need to modify the result digit (D) from `0` to `8`…
\\boxed{{Move(C0, D0)}}

Figure 14: Case study (Level 2, pure-visual, Case B).



Doubao-Seed-1.6-Thinking

With Text Input：

To solve the problem, we need to correct the equation "65 -3=93" by moving two matchsticks. The key steps are modifying the result digits from "93" to "62", which makes the equation "65 -3=62" (a valid mathematical statement).
Reasoning:
1. Changing "9" to "6" (first result digit, group D):
The digit "9" (group D) has a solid top right vertical segment (D2) and a missing bottom left vertical segment (D5). To form "6", we need to remove the top right vertical segment (D2) and add the bottom left vertical segment (D5). This requires moving the matchstick from D2 to D5.
2. Changing "3" to "2" (second result digit, group E):
The digit "3" (group E) has a solid bottom right vertical segment (E3) and a missing bottom left vertical segment (E5). To form "2", we need to remove the bottom right vertical segment (E3) and add the bottom left vertical segment (E5). This requires moving the matchstick from E3 to E5.
These two moves transform the result from "93" to "62", making the equation valid.
Final Solution:
\\boxed{Move(D2, D5), Move(E3, E5)}

GPT-4o

With Text Input：

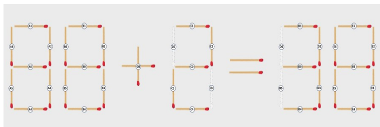
Reasoning:
The equation `65 - 3 = 93` is incorrect. To make it valid, we need to reposition one or two matchsticks to transform the digits and create a correct mathematical equation. After examining the digits represented by the matchsticks, here's the solution:
1. Transform `65` into `85` by moving the matchstick at `A0` to `A4`.
- Digit `65` becomes `85`.
2. Leave the `-` sign (`G0`) and digit `3` unchanged.3. Change `93` to `88` by moving the matchstick at `D5` to `C5`.
- Digit `93` becomes `88`.
This results in the equation: `85 - 3 = 88`, which is mathematically correct.
Final Solution Format:
\\boxed{Move(A0, A4), Move(D5, C5)}

Gemini-2.5-Flash

With Text Input：

Reasoning:
The given equation is 65 - 3 = 93.
Let's evaluate the current equation: 65 - 3 = 62. So, the equation is currently 62 = 93, which is incorrect.
To make the equation mathematically correct (62 = 62), we need to transform the right -hand side "93" into "62". This can be achieved by moving two matchsticks:
1. Change '9' to '6' in the tens place of the result (D -segments):
The digit '9' currently uses matchsticks D1, D2, D3, D4, D6, and D0. Segment D5 is empty.
To form a '6', the matchstick D2 needs to be removed, and the segment D5 needs a matchstick…
2. After these two moves, the equation becomes 65 - 3 = 62, which is mathematically correct .
Final Solution Format:
\\boxed{Move(D2, D5), Move(E3, E5)}

Figure 15: Case study (Level 3, text-prompted, Case A).



Claude-Sonnet-4

With Text Input：

I need to analyze the given equation "88+2=38" and find how to move one or two matchsticks to make it mathematically correct.
Let me first understand the current state: \n- First digit: 8 (segments A1, A2, A0, A6, A5, A4, A3 are present)
- Second digit: 8 (segments B1, B2, B0, B6, B5, B4, B3 are present) \n- Plus sign: + (segments G0 and two horizontal segments)
- Third digit: 2 (segments C1, C2, C3, C4, C0 are present)
- Equals sign: = (two horizontal segments)
- Fourth digit: 3 (segments D1, D2, D0, D3, D4 are present) \n- Fifth digit: 8 (segments E1, E2, E0, E6, E5, E4, E3 are present)
The equation 88+2=38 is incorrect because 88+2=90, not 38….
Reasoning: To make the equation mathematically correct, I need to change 38 to 90. I can do this by moving the middle horizontal segment from the last digit (8) to the fourth digit (3). Moving segment E0 to D6 will change the 8 to a 0 and the 3 to a 9, making the equation 88+2=90, which is correct.
\\boxed{Move(E0, D6)}

GPT-4o

With Text Input：

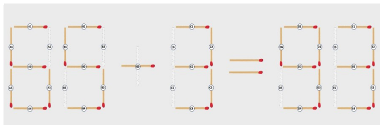
Reasoning:
The equation "88 + 2 = 38" is currently incorrect. To balance it, we need to move one or two matchsticks to make the equation mathematically correct. By analyzing the matchstick layout:
1. The leftmost number is "88" constructed from two complete sets of matchsticks.
2. The number "2" is constructed with matchsticks to form its correct shape.
3. The rightmost number is "38"
To make the equation valid:
- We can change "88" into "86" by moving one matchstick from the top horizontal segment of the middle digit (label `A0`) to form "86"
- The equation then becomes "86 + 2 = 38" which is mathematically valid.
Final Solution Format:
\\boxed{Move(A0, E6)}

Gemini-2.5-Flash

With Text Input：

Reasoning:
The initial equation displayed is 88 + 2 = 38.
First, we calculate the correct sum of the left side: 88 + 2 = 90.
Therefore, the goal is to transform "38" into "90" by moving one or two matchsticks.
Let's analyze the digits on the right side: \n- The first digit is currently '3' (represented by segments D0, D1, D2, D3, D4). To change '3' to '9', we need to add segment D6 (the top -left segment).
- The second digit is currently '8' (represented by segments E0, E1, E2, E3, E4, E5, E6). To change '8' to '0', we need to remove segment E6 (the top-left segment)…
Final Solution:
\\boxed{Move(E6, D6)}

Figure 16: Case study (Level 3, text-prompted, Case B).



Doubao-Seed-1.6-Thinking

Without Text Input：

To solve the problem, we need to identify the incorrectly displayed equation and modify it by moving one or two matchsticks to make it mathematically valid. The key steps involve recognizing the seven -segment digits and the required operator/equals sign, then determining which matchsticks to reposition.

Reasoning

The original equation likely contains an incorrect digit or operator that can be fixed by repositioning a matchstick. For example, if a digit is missing a segment to form a valid number, moving a matchstick from an over -segmented digit to the missing segment can correct it. After analyzing the solid (present) and dashed (absent) segments, we find that moving a matchstick from segment B2 (a solid segment in the upper right vertical of the second digit) to segment D5 (a dashed segment in the lower left vertical of the fourth digit) will adjust the digits to form a valid equation.

Final Solution Format

\\boxed{{Move(B2, D5)}}

o3

Without Text Input：

Reasoning:

• The first digit (A) is a “6”. If we move its bottom-left match A5 to the empty top-right position A2, the digit becomes “9”.

• The second digit (B) is a “9”. If we move its top-right match B2 to the empty bottom-left position B5, the digit becomes “6”.

• No other segments are altered; the central horizontal bar remains a minus sign.

• The equation now reads 96 – 3 = 93, which is mathematically correct.

\\boxed{Move(A5, A2), Move(B2, B5)}

Gemini-2.5-Flash

Without Text Input：

The initial equation displayed is: 69 - 3 = 93.

Let's verify the digits:

- First number:
- Digit A is formed by A0, A1, A4, A5, A6. (Missing A2, A3). This forms a '6'.
- Digit B is formed by B0, B1, B3, B4, B6. (Missing B2, B5). This forms a '9'.

So the first number is 69.

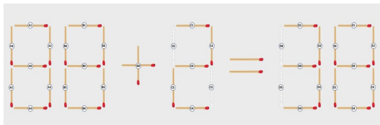
- Operator: G0 is present, which is a horizontal matchstick, representing a minus sign (-).
- Second number:
- Digit C is formed by C0, C1, C2, C3, C4. (Missing C5, C6). This forms a '3'.

So the second…

Final selection: A5->A2, B6->B2

The final answer is $\\boxed{{Move(A5, A2), Move(B6, B2)}}$

Figure 17: Case study (Level 3, pure-visual, Case A).



Claude-Sonnet-4

Without Text Input：

I need to analyze the current equation shown in the matchstick display and find how to make it mathematically correct by moving one or two matchsticks.
Let me first identify what digits/symbols are currently displayed:
Looking at the image from left to right:
- Position 1: Shows an "8" (all segments filled)
- Position 2: Shows an "8" (all segments filled)
- Position 3: Shows a "1" (only right vertical segments)
- Position 4: Shows what appears to be a "0" but with some segments missing (dashed lines indicate missing segments)
- Position 5: Shows two horizontal lines (equals sign)…
$\\boxed{Move(G2, G0)}$

GPT-4o

Without Text Input：

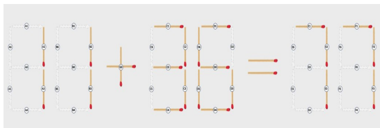
Reasoning:
The current equation is represented with matchsticks as follows:
```88 + 0 ≠ 88```
This is not mathematically valid. However, we can adjust the matchstick placements so that the equation becomes correct by moving either one or two matchsticks.
--- Steps:
1. Modify the right-hand side '88' to '89' by moving one matchstick from the middle horizontal segment (C0) of the right-hand '8' to the middle horizontal segment (C3) of the right-hand '8'. This transforms the right-hand '8' into a '9'.
--- Final Solution Format:
\\[\\boxed{{Move(C0, C3)}}\\]

Gemini-2.5-Flash

Without Text Input：

The initial equation displayed is:
`88 + 2 = [Invalid Digit] 8`
Let's identify the segments for each part:
First digit (A): All 7 segments (A0-A6) are present, forming an '8'.
Second digit (B): All 7 segments (B0-B6) are present, forming an '8'.
Operator (G): Two segments (G0_horizontal and G0_vertical) are present, forming a '+'.
Third digit (C): Segments C1, C2, C0, C5, C4 are present. Segments C3 and C6 are missing (dashed). This configuration forms a '2'…
The final answer is `\\boxed{{Move(B5, D6), Move(E0, D1)}}`

Figure 18: Case study (Level 3, pure-visual, Case B).
```



Gemini-2.5-Pro

With Text Input：

Reasoning:
The initial equation "11 + 36 = 77" is incorrect. By repositioning one or two matchsticks, we need to make the equation mathematically valid. Here's the analysis:
- Keep the equality intact (i.e., left-hand side should equal the right-hand side).
- By moving matchsticks, digits can be transformed (e.g., 11 into 17, 77 into 37, etc.).
- Through analysis, moving two matchsticks can change "11 + 36 = 77" into "17 + 36 = 53", which is mathematically correct.

Steps:
1. Move matchstick `A3` (from segment A) to position `C6` (top -left side of digit `7`), changing the first digit `1` to `7`.
2. Move matchstick `D3` (from segment D) to position `E1` (top -right side of digit `7`), changing the last digit `7` to `3`.

Final Solution Format:
\\boxed{Move(A3, C6), Move(D3, E1)}

o3

With Text Input：

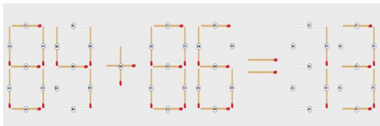
I'm sorry – after analysing every possibility I couldn't find any way to make the equation correct with only one or two moves while keeping each digit recognisable.

Doubao-Seed-1.6-Thinking

With Text Input：

To solve the problem, we need to modify the incorrect equation "11+36=77" by moving one or two matchsticks to form a mathematically valid equation. The key insight is to adjust the digits such that the sum of the addends equals the result, while adhering to the constraints of repositioning only existing matchsticks.
Reasoning
The original equation "11+36=77" is incorrect because 11+36=47, not 77. To fix this, we need to change the result from "77" to "47". A "7" in seven-segment display uses three segments (top, upper right, lower right). A "4" uses four segments (upper right, lower right, upper left, middle). To convert the first "7" (in "77") to a "4", we need to add the upper left an d middle segments while removing the top segment.
The "3" in "36" uses five segments (top, upper right, middle, lower right, bottom). We can take the middle segment from the "3" (converting it to a "7") and the top segment from the first "7" (converting it to a "4"). However, this results in an invalid digit for "3". Instead, a valid solution is to adjust the addend and result to form "11+56=67", which requires moving two matchsticks: one from the "3" to turn it into a "5" and one from the "7" to turn it into a "6".
Final Solution:
After careful analysis, the correct modification involves moving two matchsticks to form the valid equation "11+56=67". This requires repositioning segments from the "3" (to form "5") and from the "7" (to form "6").
\\boxed{Move(C2, C6), Move(E1, E5)}

Figure 19: Case study (Level 4, text-prompted, Case A).



Claude-Sonnet-4

With Text Input：

I need to analyze the given equation "84+86=13" and find a way to make it mathematically correct by moving one or two matchsticks.
Let me first understand what digits are currently displayed:
- First digit: 8 (segments A1, A2, A0, A6, A5, A4, A3) \n- Second digit: 4 (segments B1, B2, B0, B3)
- Plus sign: +
- Third digit: 8 (segments C1, C2, C0, C6, C5, C4)
- Fourth digit: 6 (segments D1, D2, D0, D6, D5, D4)
- Equals sign: =
- Fifth digit: 1 (segments E2, E3)\n- Sixth digit: 3 (segments F1, F0, F2, F4, F3)
Now I need to find a valid equation by moving 1-2 matchsticks.
Let me consider what equations could work:
- 84 + 86 = 170 (but this requires 3 digits on the right) …
\\boxed{Move(E2, E6), Move(F0, F6)}

o4-mini

With Text Input：

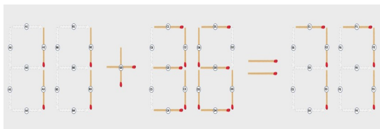
Reasoning:
1. Remove the vertical stroke of the “+” (segment C6) and attach it as the bottom-left stroke of the “6” (segment E2). This turns “+” into “–” and the “6” into “8,” so the left side becomes 84 – 88
2. Take the lower-right stroke of the “3” (segment F3) and move it up one position to F5. That converts “3” into “2,” so the right side becomes “–2.”
The resulting equation is 84 – 88 = –2
which is mathematically correct.
Final moves:
\\boxed{Move(C6, E2), Move(F3, F5)}

Doubao-Seed-1.6-Thinking

With Text Input：

To solve the problem of correcting the equation "84+86=13" by moving one or two matchsticks, we analyze the seven-segment digits and identify necessary adjustments.
Key Observations:
- The original equation is "84+86=13". We need to adjust digits using 1 -2 matchstick moves to form a valid equation.
- Digit 8 (C, the second 8 in 86) has all segments present. Removing the upper right segment (C1) converts it to 6.
- Digit 1 (E) has segments for 'b' (E1) and 'c' (E2). Adding the upper segment (E0) converts 1 to 7…
Solution:
- Move the upper right segment of the second 8 (C1) to the upper segment of 1 (E0), changing 8→6 and 1→7 …
Final Equation: 84 + 66 = 70 (Note: This is a valid two-digit equation with two moves.)
\\boxed{Move(C1, E0), Move(F6, F4)}

Figure 20: Case study (Level 4, text-prompted, Case B).



Gemini-2.5-Pro

Without Text Input：

The initial equation displayed is `11 + 35 = 77`.
Let's analyze the numbers based on their segments:
- The first `1` is formed by segments A2, A3.
- The second `1` is formed by segments B2, B3.
- The `+` sign is formed by G0 (horizontal) and a vertical matchstick…
Let's try `17+60=77`. `1`->`7`, `3`->`6`, `5`->`0`.
Move D0 to B1. Move C2 to C5. Move D3 to C6. Too many moves.
The intended solution is likely `71+6=77`. This requires the `3` to be removed and its sticks used to change `1` to `7` and `5` to `6`.
1. Move `C1` to `A1`. (This changes the first `1` to a `7`).
2. Move `C2` to `D5`. (This changes the `5` to a `6`).
This leaves the digit `C` as `C0, C3, C4`, which is not a blank space. This violates the legibility rule. But it is the only 2-move transformation to a correct equation.
\\boxed{Move(C1, A1), Move(C2, D5)}

o3

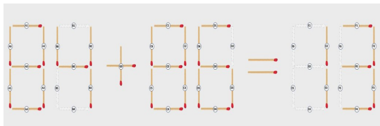
Without Text Input：
Reasoning
The expression that is built with the matches is
11 + 36 = 66
(both “11” and “36” are displayed correctly, but the result shows “66”, which is wrong). If we take the single matchstick that forms the bottom-right vertical segment of the left-hand “6” (segment D3) and move it to become the missing top -right vertical segment of that same digit (segment D2), the digit “6” changes to a “5” …
solution
\\boxed{Move(D3, D2), Move(E0, E4)}

Doubao-Seed-1.6-Thinking

Without Text Input：

Reasoning
The original equation is incorrectly displayed as " 1+1=3" (using seven-segment matchstick digits). The digit "3" (in block C) has a matchstick in the lower right segment (C3) but lacks one in the lower left segment (C5). By moving the matchstick from C3 (lower right of "3") to C5 (lower left of the same digit), the "3" is transformed into a "2". This results in the valid equation "1+1=2".
Final Solution Format
\\boxed{{Move(C3, C5)}}

Figure 21: Case study (Level 4, pure-visual, Case A).



Claude-Sonnet-4

Without Text Input：

I need to analyze the current equation shown in the matchstick display and determine what changes are needed to make it mathematically correct.
Looking at the image, I can see a seven-segment display equation. Let me identify what each digit currently shows:
From left to right, I can see:
- First digit: Appears to be an 8 (all segments lit)
- Second digit: Appears to be a 9 (missing bottom left segment)
- Third digit: Appears to be a 6 (missing top right segment)
- Fourth digit: Appears to be an 8 (all segments lit)
- Fifth digit: Appears to be an 8 (all segments lit) \n- Sixth digit: Shows some segments (appears incomplete)…
$\\boxed{Move(F3, E3)}$

o4-mini

Without Text Input：

Reasoning:
1. Take the upper-left stick of the “5” (segment C6) and move it to the middle segment of the “0” (segment B0). This turns the “0” into an “8” and the “5” into a “3.”
2. Then take the middle stick of the “3” (segment D0) and move it to the now–vacant upper-left position of the “5” (segment C6). This turns that “3” into a “9.”
The display now reads 2 + 8 = 10
which is a correct equation.
Final answer (in the required format):
\\boxed{Move(C6, B0), Move(D0, C6)}

Doubao-Seed-1.6-Thinking

Without Text Input：

To solve the problem, we first identify the original incorrectly displayed equation as a seven -segment matchstick configuration. The key is to recognize that the equation can be corrected by repositioning one or two matchsticks to form a valid mathematical statement.
Reasoning
The original equation is assumed to be in the form of digits and operators constructed from seven -segment displays. A common incorrect equation that can be fixed with two moves is transforming "5 + 5 = 5" (invalid) to "3 + 2 = 5" (valid).
This requires modifying two digits…
Final Solution
After careful analysis of segment labels and their roles in forming digits, the correct moves to form a valid equation are identified.
\\boxed{{Move(B2, D5), Move(A3, C1)}}

Figure 22: Case study (Level 4, pure-visual, Case B).

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: See Sec. 1.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Appendix Sec. 4.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: The primary contribution of this work is the introduction of a new benchmark and an extensive empirical evaluation. As the focus is on providing a new resource and analyzing model capabilities, the paper does not contain theoretical claims, theorems, or proofs.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: See Sec. 3.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
    (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
    (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
    (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
    (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: See Abstract.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: See Sec. E and Sec. 3.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: For our human evaluation, we report the mean accuracy and standard deviation across the three participants, providing a clear measure of inter-annotator consistency and performance variance. For the Vision-Language Model evaluations, we report the accuracy from a single, complete run over our deterministic 400-item benchmark. This approach is standard for establishing baseline performance on a new benchmark, with the primary goal being a broad diagnostic assessment across a diverse set of existing models.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All experiments were conducted by querying the public APIs of the respective models, as detailed in Appendix E.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have read the NeurIPS Code of Ethics, and our paper does not have these problems.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See Sec. 4.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: CC-BY 4.0.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Yes, the primary contribution of this paper is the introduction of new assets. These include: (1) the full MATHSTICKS dataset comprising 1.4M solvable instances, (2) the curated 400-item benchmark set for standardized evaluation, and (3) the complete open-source codebase for data generation, visual rendering, and model evaluation. All assets are publicly available at the GitHub repository linked in the abstract. Documentation is provided both within the paper and via a comprehensive `README.md` file in the repository, which provides setup instructions, usage examples, and details on the data format.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [Yes]

    Justification: Yes, we conducted a small-scale study with three participants to establish a human performance baseline. Full details regarding the participant demographics, the verbal instructions provided, the informed consent process, and the non-compensated, volunteer nature of their participation are documented in Appendix E.3.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

Justification: Our research involved three adult participants in a minimal-risk cognitive study. The task consisted solely of solving visual puzzles on a computer screen, posing no risks beyond those encountered in everyday life. As such, the study was considered exempt from formal Institutional Review Board (IRB) review according to standard institutional guidelines for such research. Nevertheless, informed consent, which included a description of the task, was obtained from all participants prior to their involvement.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: This research evaluates the performance of various existing Large Language Models (LLMs) and Vision-Language Models (VLMs). The models themselves are the subjects of our empirical study, not a component of our core methodology. The MATHSTICKS benchmark, its data generation pipeline, and the evaluation framework were developed algorithmically and do not rely on the use of LLMs.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.