Can Large Language Models Correctly Interpret Equations with Errors?

Lachlan McGinness[1,2] and Peter Baumgartner[2,1]

[1] Australian National University
[2] CSIRO/Data61, Australia

Abstract. This paper explores automated grading as a potential application of AI for education practitioners in the Australian Physics Olympiads. Large Language Models were used to extract equations from student responses and convert these into a standardised format for a computer algebra system. Models with more than fourteen billion parameters were unable to complete the task in the required timeframe. No open source model was able to achieve the desired level of accuracy given resource constraints available for marking the exam. To improve the accuracy, we implement LLM-modulo and consensus frameworks and report on the results. Future work to improve performance could involve breaking the task into smaller components before parsing to the models.

Keywords: Neuro-Symbolic AI · Automated Grading · Physics.

1 Introduction

The Australian Physics Olympiad is an annual physics exam in Australia with approximately fifteen hundred 16-18 year-old participants. For the last five years the exam was conducted online with a combination of question types, where students are allowed to submit their responses either in a typed format or by scanning hand-written working. Students sit the test in early August and it is graded by a team of approximately fifteen markers in a 48-hour period two weeks later.

In this paper we explore the potential of applying automated grading to mark a sample of 200 typed responses to one question of the exam. We evaluate the performance of Large Language Models (LLMs) to correctly extract equations from a student response and write them in a standardised format.

Specifically, we evaluate the LLM-Modulo framework where the LLM is included as a part of a feedback loop for error correction and improving results [9]. The LLM-Modulo framework was originally designed for the application of planning problems. Our contribution is to apply this framework to physics marking problems.

2 Background

Physics problems are very difficult to mark automatically because they require a wide range of inputs including numerical responses (often with units

and directions), short answers, algebraic expressions, diagrams and plots of data/functions. Recent attempts include using machine learning [13] or Large Multimodal Models (LMMs) [10]. Although this method has shown to be somewhat successful [11], there are a number of ethical considerations which have been ignored.

To be deployed in Australia, an AI technology should adhere to the Australian Government's AI Ethics Principles [17]. These include environmental well-being, transparency and privacy protection. To address these principles, a local, open-source LLM could be used for feature extraction to make the process transparent and enhance privacy by keeping data on a local system. If there is an error in the LLM feature extraction, this can be easily seen, contested and corrected by a student or teacher. Local models make the teachers aware of the compute that they are using, exposing the normally hidden electricity usage and environmental cost.

The AI Ethics Principles also include reliability and contestability. LLM reasoning about algebraic expressions is unreliable. A rigorous source of reasoning over algebraic formulas is given by the field of automated theorem proving. Automated Theorem Provers are grounded in mathematical logic and are able to produce indisputable proofs. In this paper, we use the Z3 SMT (Satisfiability Modulo Theory) prover [4]. We are also using the computer algebra system SymPy for the same purpose [15].

Our proposed pipeline that meets Australia's AI Ethics Principles is called AlphaPhysics. The overall pipeline structure can be seen in Figure 1. Although there is no official connection to Google Deepmind, AlphaPhysics follows the same structure as AlphaGeometry where a neural model guides a symbolic deduction engine to solve Olympiad geometry problems [20].

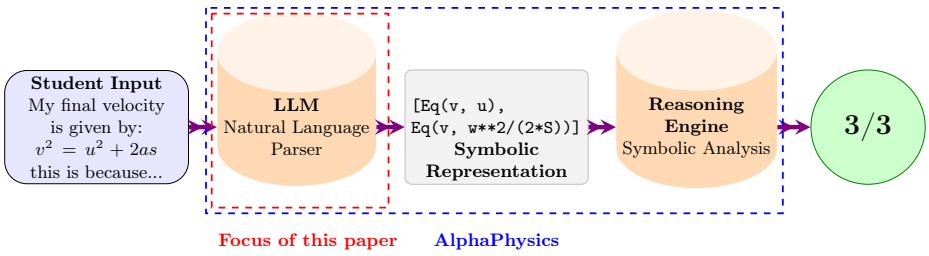

Fig. 1. The AlphaPhysics Pipeline. The student response is parsed by an LLM. A symbolic reasoning engine then determines the student's grade.

In this paper, we do not explore the limitations of computer algebra systems, but instead focus on the first half of this pipeline: using an LLM for feature extraction. We perform experiments to evaluate the performance of open source models on extracting equations from student responses. In the following sections

we describe our experiments, their results, and then describe the key areas for future work.

3 Methods

As AlphaPhysics is a system which is designed to be run on a local system, only small LLMs can be used. Therefore in our experiments we investigate techniques which could improve the accuracy of an LLM without increasing its size:

- **Direct Method** - where a model is given a prompt to extract equations from a student response.
- **LLM-Modulo SymPy** - after the Direct Method is applied, the model is warned if its response does not match the required syntax. The model is then given a chance to improve its response. See Algorithm 3.1.
- **LLM-Modulo Z3** - similar to the LLM-Modulo SymPy approach, but the model is given specific feedback for the type of syntactic error. For example "symbol 'U' is undefined".
- **Consensus** - builds on the LLM-Modulo SymPy procedure by comparing two models' responses using Z3. If they do not match then the models are given three attempts to reach consensus, see Algorithm 3.2.

The sample of 200 responses contained 30 blank responses which were parsed automatically without calling an LLM. Each technique was tested with a range of models, as described in Section 3.1, using the metrics described in Section 3.2.

Algorithm 3.1: LLM_Modulo Algorithm: gives LLM feedback

```
Input: student_answer, max_attempts, parser
Output: modulo_list ;                    // parsed equations or final best attempt
direct_list ← Get_llm_direct_response(student_answer);
correct_syntax, error_messages ← Attempt_parse(direct_list, parser);
if correct_syntax = True then
    modulo_list ← direct_list ;          // All equations syntactically correct.
    return modulo_list;
attempts ← 0;
modulo_list ← direct_list;
while attempts ≤ max_attempts do
    attempts ← attempts + 1;
    modulo_list ← Get_llm_repair_response(student_answer, modulo_list,
      error_messages);
    correct_syntax, new_error_messages ← Attempt_parse(modulo_list);
    if correct_syntax = True then
        break;
    error_messages ← error_messages ∪ new_error_messages;
return modulo_list;
```

The Algorithms B.1 (attempt_parse), B.2 (check_equivalence) and B.3 (Z3 _Equivalent) are helper functions used in LLM_Modulo and Consensus, see Appendix B. The core of these functions checks for equivalence and entailment

as follows. For two lists of equations E_1 and E_2, we say that E_1 entails E_2 iff for every e_2 in E_2 there is an e_1 in E_1 such that e_1 entails e_2. We say that E_1 is equivalent to E_2 if E_1 entails E_2 and E_2 entails E_1. All reasoning is carried out with respect to a background theory, real arithmetic, which is natively supported by Z3.

As an example, consider the following two lists of equations $[v = u + w, v = u + w]$ and $[v - u = w, v = u - w]$ which we call E_1 and E_2 respectively. E_2 entails E_1 as the first formula in E_2 entails both formulas in E_1. However E_1 does not entail E_2 as $v = u - w$ is not entailed by any formula in E_1. Therefore the two lists are not equivalent.

Algorithm 3.2: Consensus_Algorithm: LLM Equation Agreement

```
   Input: Model1, Model2, student_response, max_attempts, parser
   Output: Model1_equations, Model2_equations
 1 Model1_equations ← LLM_Modulo(student_response, parser, Model1);
 2 Model2_equations ← LLM_Modulo(student_response, parser, Model2);
 3 attempts ← 0;
 4 while attempts ≤ max_attempts do
 5 |   attempts ← attempts + 1;
 6 |   are_equivalent ← Check_equivalence(Model1_equations, Model2_equations);
 7 |   if are_equivalent = True then
 8 |   |   return Model1_equations, Model2_equations;
 9 |   consensus_prompt ← Create_consensus_prompt(Model1_equations,
   |     Model2_equations, student_response);
10 |   Model1_equations ← Get_LLM_response(consensus_prompt, Model1);
11 |   Model2_equations ← Get_LLM_response(consensus_prompt, Model2);
12 return Model1_equations, Model2_equations;
```

We also use equivalence to compare the LLM extracted lists of equations to those provided by the examiners as a measurement of accuracy. Student working is often messy, out of order or contains repeats; however most examiners do not wish to penalise students for this. Our definition for equivalence allows for this, as long as two lists of equations contain the same information, a repetition or change in order is not considered a difference.

3.1 Selected Models

In this experiment we consider only open source models, see Table 3 in Appendix A. Some of these models are thinking models and some were trained by distilling a larger model. Thinking models are LLMs which are specifically trained to produce a verbose set of thinking tokens before producing their final answer tokens. Distillation is a technique where a smaller model is trained to reproduce the behaviour of a larger one [18].

Our experiments mostly focus on smaller models that can be run on consumer-grade hardware. All models with less than seventy billion parameters were run on a local machine with an Intel Core-i9-13900K CPU (3-5.8GHz), 64GB of DDR5 (4800MT/s) and an NVIDIA GeForce RRTX 4060Ti GPU with 16GB of dedicated GPU memory.

3.2 Evaluation of Models and Techniques

To evaluate each model and technique we collected each of the following pieces of information:

- **Wall Time** - Measure of computational expense and therefore cost, environmental impact as well as practicality. Wall time was not collected for models which could not be run on local systems as it is not an accurate measure of the computational expense nor comparable with other models.
- **Number of Tokens Generated** - This is primarily used to better understand the wall time measurements for thinking/non-thinking models.
- **Accuracy** - LLMs were asked to provide their equations in SymPy format. Each of the responses was converted from SymPy format to Z3 to verify the equivalence of the LLM list of equations to the markers' list of equations (see Algorithm B.2). The portion of problems where the two lists were found to be equivalent was reported as the accuracy score.

4 Results

The raw data from the original experiments are shown in Table 1. After the initial data were taken, targeted experiments with distilled models were performed, see Table 4 in Appendix C. It is clear from Table 1 that the LLM-Modulo SymPy (simply telling a model it is wrong without explanation) makes almost no improvement to performance. Figure 2 shows that as model size increases so does performance.

Table 1. Raw data for each model and each technique.

Model	Llama 3.1	Deepseek Llama 3.1	Deepseek	Phi4	Deepseek	Mathstral	Deepseek Llama 3.2		Gemma	Deepseek Llama 3.2		
Model Size (B)	405	672	70	70	14	14	7	7	3	2.2	1.5	1
Quantisation (bit)	16	16	16	16	16	16	16	16	16	16	16	16
Direct												
Time (s)	NA	NA	NA	NA	4,460	90,001	570	17,141	209	183	2,946	146
Tokens	5,705	105,836	5,257	109,815	5,582	117,626	6,625	132,484	6,740	6,716	175,361	10,383
Correct Answers	154	169	149	166	141	129	80	110	75	53	80	33
LLM-Modulo SymPy												
Time (s)	NA	NA	NA	NA	4,604	91,010	673	17,422	211	321	3,024	156
Tokens	5,900	107,884	5,326	112,575	5,769	118,928	7,893	134,657	6,776	12,059	179,960	10,978
Correct Answers	164	169	149	166	141	130	80	110	75	53	80	33
After Consensus	176	176	156	173	140	132	85	98	72	57	82	71
LLM-Modulo Z3												
Time (s)	NA	NA	NA	NA	8,278	128454	885	22,447	220	438	3,508	188
Tokens	7,146	122,038	6,496	116,469	7,085	114597	10,387	151,793	6,838	15,818	208,440	13,144
Correct Answers	169	173	151	167	146	143	85	123	74	54	89	61

Figure 2 demonstrates that small models receive greatest improvement when given feedback about syntax, especially thinking models. On the other hand, larger models gain more benefit from discussion and checking their answers with others. Figure 3 shows that the consensus process greatly increases computational expense with only a small increase to overall accuracy.

Here we provide an order of magnitude estimate for the maximum time that is feasible for marking the Australian Physics Olympiad exam. Normally the

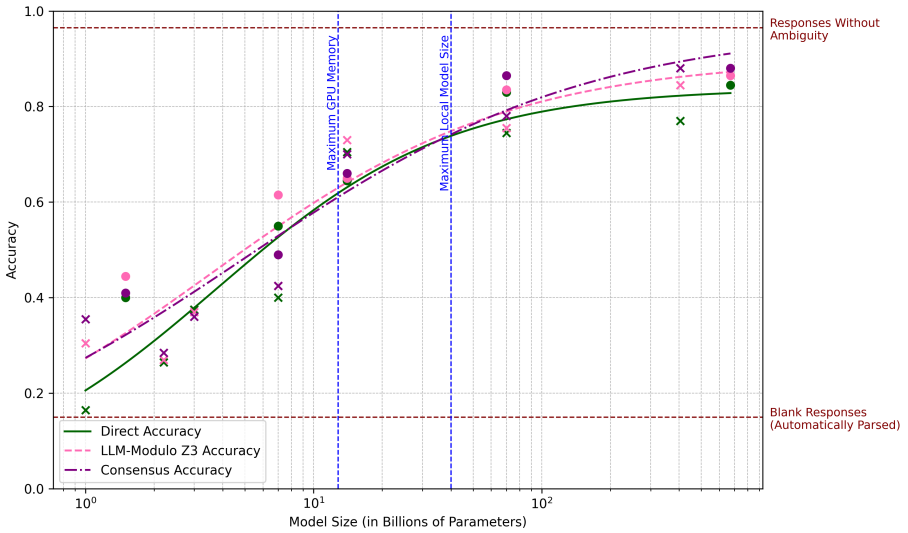

Fig. 2. Plot of accuracy against model size. Colour indicates the technique being used. Circles correspond to thinking models (like Deepseek) while crosses refer to other models. Blue vertical lines indicate the maximum model size that fits in the GPU memory and the maximum model size which can be run on the hardware. Sigmoid functions show the overall trend of the data.

marking occurs in a 48 hour period. This could be extended to approximately two weeks for the 1500 responses to 45 questions. This means that a sample of 200 responses to one question (as used in this experiment) should take approximately 4000 seconds to mark (20 seconds per response). This includes all steps, not just the pre-processing explored in this paper. The time could be scaled up by a factor of two or three by getting multiple computers and further extending the time period of marking, but 4000 seconds is the approximate time requirement. This time limit is shown as a blue vertical line on Figure 3.

The errors that were made by the models with the highest accuracy were investigated manually and classified into error types, see Table 2. Models most commonly made errors when students made reference to an attachment that did not exist, in these cases the model was likely to hallucinate equations.

5 Discussion and Future Work

In this section, the results of the experiments will be related to the practical challenges of marking Australian Physics Olympiad papers in the required timeframe. This is not meant to signal that the Australian Physics Olympiad is considering the use of Automated Marking tools for future exams.

Figure 2 shows that the highest accuracy score achieved by any model and condition was 88%. This is 8.5% below the desired threshold of 96.5% where the

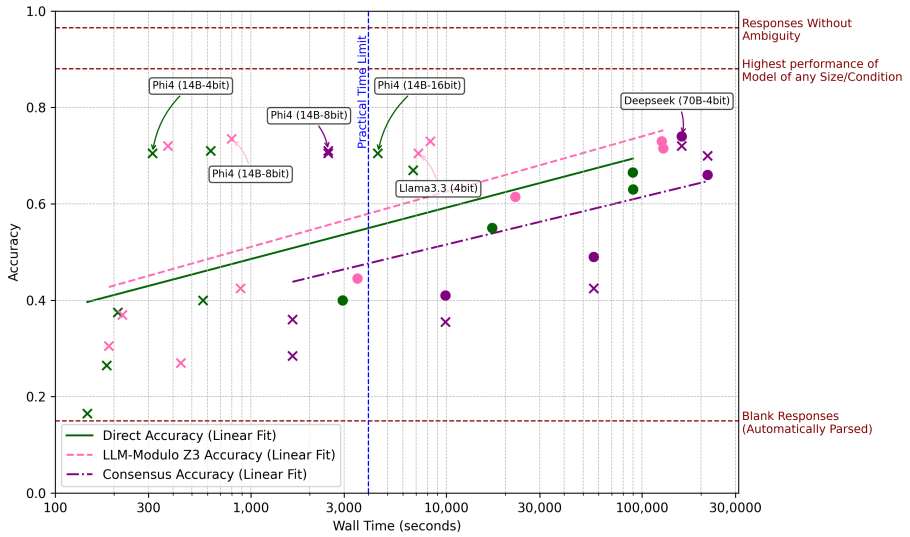

Fig. 3. Model Accuracy vs. Time to Run. Colour is used to indicate the technique. Circles correspond to thinking models (DeepSeek), all other models have crosses.

Table 2. Frequency of Different Types of Errors for the Largest Models after reaching consensus.

Type of Error	Frequency of Error
Fictitious attachment	8
Over-interpreted student answer	5
Under-interpreted student answer	4
Incorrect capitalisation of variables	4
Problem contained incorrect variables	4
Used undefined symbols	3
Missed equations in a paragraph	3
Student referred to working not there	1
Total Errors	32

markers find the responses ambiguous. Therefore further increases in accuracy would be required for a model to be sufficiently reliable to be included in the marking pipeline of the Australian Physics Olympiad Exam. One key area for future work would be to determine new model architectures or strategies to improve accuracy. One strategy which has been shown to work is to break the task into smaller components [14], which could be implemented by including only one line of the student response per model call instead of the entire response.

Figure 3 shows that some of the models are able to respond quickly enough that it would be feasible to use them grading. However, the thinking models which make use of excessive tokens run too slowly to be useful on the time-scale of the Australian Physics Olympiad exam. These preliminary experiments

indicate that the maximum feasible model size is approximately 14 billion parameters. Phi4 shows the most promise of these models and decreasing the size of the floating point numbers from 16 bits to 4 bits only marginally decreases performance. Nearly all of the improvements to LLMs since December 2023 can be attributed to increasing the computational expense through either model size or number of tokens (chain of thought or thinking models). A new approach to improving model performance will be required in order to meet the speed and hardware requirements, some recent advancements include diffusion-based language models [12,16] or controlling test time compute using a recurrent block [7].

In this work we only consider typed responses for algebraic expressions. Future work could consider other types of questions such as diagrams, plots, numerical responses and short answer questions. As students are allowed to submit hand-written responses, another key area for future work is to the use of LMMs to parse these.

6 Conclusions

LLM-modulo frameworks were only found to be effective if they provide an LLM with specific feedback on how to improve its response. Thinking models such as DeepSeek were too computationally expensive, regardless of their size because of the large number of tokens they produce.

Microsoft's Phi4, a fourteen billion parameter model, was fast enough to meet time restrictions with only marginal decreases in performance when using four bit and eight bit floating point numbers. Future work will explore other input types and new methods to increase this accuracy with minimal impact to runtime.

Disclosure of Interests. The first author of this paper is a staff member in the Australian Physics Olympiad program and a high school physics teacher.

References

1. Abdin, M.I., Zhang, Y.: Phi-4 Technical Report (Dec 2024), https://www.microsoft.com/en-us/research/publication/phi-4-technical-report/
2. AI, M.: Llama 3.2 | Model Cards and Prompt formats (2024), https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_2/
3. AI, M.: Llama 3.3 | Model Cards and Prompt formats (2024), https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_3/
4. De Moura, L., Bjørner, N.: Z3: an efficient SMT solver. In: Theory and practice of software. pp. 337–340. TACAS'08/ETAPS'08 (Mar 2008)
5. DeepSeek-AI, Bi, X., Zou, Y.: DeepSeek LLM: Scaling Open-Source Language Models with Longtermism (Jan 2024). https://doi.org/10.48550/arXiv.2401.02954, http://arxiv.org/abs/2401.02954, arXiv:2401.02954 [cs]
6. DeepSeek-AI, Guo, D., Gao, Z.: DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning (Jan 2025). https://doi.org/10.48550/arXiv.2501.12948, http://arxiv.org/abs/2501.12948, arXiv:2501.12948 [cs]

7. Geiping, J., McLeish, S., Jain, N., Bhatele, A., Goldstein, T.: Scaling up Test-Time Compute with Latent Reasoning: A Recurrent Depth Approach (2025). https://doi.org/10.48550/arXiv.2502.05171, arXiv:2502.05171 [cs]
8. Grattafiori, A.: The Llama 3 Herd of Models (Nov 2024). https://doi.org/10.48550/arXiv.2407.21783
9. Kambhampati, S., Valmeekam, K., Guan, L., Verma, M., Stechly, K., Bhambri, S., Saldyt, L.P., Murthy, A.B.: Position: LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks. In: ICML. pp. 22895–22907. PMLR (Jul 2024), https://proceedings.mlr.press/v235/kambhampati24a.html
10. Kortemeyer, G.: Toward AI grading of student problem solutions in introductory physics: A feasibility study. Physical Review Physics Education Research **19**(2), 020163 (Nov 2023)
11. Kortemeyer, G.: Performance of the pre-trained large language model GPT-4 on automated short answer grading. Discover Artificial Intelligence **4**(1), 47 (Jul 2024)
12. Li, X.L., Thickstun, J., Gulrajani, I., Liang, P., Hashimoto, T.B.: Diffusion-LM improves controllable text generation. In: Neural Information Processing Systems. pp. 4328–4343 (2022)
13. Mcginness, L., Baumgartner, P.: CON-FOLD Explainable Machine Learning with Confidence. Theory and Practice of Logic Programming **24**(4), 663–681 (Jul 2024). https://doi.org/10.1017/S1471068424000346
14. McGinness, L., Baumgartner, P., Onyango, E., Lema, Z.: Highlighting Case Studies in LLM Literature Review of Interdisciplinary System Science. In: AI 2024: Advances in Artificial Intelligence. pp. 29–43 (2024). https://doi.org/10.1007/978-981-96-0348-0_3
15. Meurer, A., Smith, C., Paprocki, M., Scopatz, A.: SymPy: symbolic computing in Python (2024), https://www.sympy.org/en/index.html
16. Nie, S., Zhu, F., You, Z., Zhang, X., Ou, J., Hu, J., Zhou, J., Lin, Y., Wen, J.R., Li, C.: Large Language Diffusion Models (2025). https://doi.org/10.48550/arXiv.2502.09992, arXiv:2502.09992 [cs]
17. Resources, D.o.I.S.a.: Australia's AI Ethics Principles (Oct 2024), https://www.industry.gov.au/publications
18. Sanh, V., Debut, L., Chaumond, J., Wolf, T.: DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter (Mar 2020). https://doi.org/10.48550/arXiv.1910.01108, http://arxiv.org/abs/1910.01108, arXiv:1910.01108 [cs]
19. Team, M.A.: MathΣtral | Mistral AI (2024), https://mistral.ai/news/mathstral
20. Trinh, T.H., Wu, Y., Le, Q.V., He, H., Luong, T.: Solving olympiad geometry without human demonstrations. Nature **625**(7995), 476–482 (Jan 2024). https://doi.org/10.1038/s41586-023-06747-5, https://www.nature.com/articles/s41586-023-06747-5

A Selected Models

Table 3 classifies the Large Language Models used in the experiments into thinking or non-thinking and by number of parameters.

Table 3. List of Open Source models used in the experiments.

Model Size (Number of Parameters)	**Non-thinking Model**	**Thinking Model**
1-2B	Llama3.2 1B [2] A small model released by MetaAI in 2024	DeepSeek 1.5B [6] A Qwen distilled version of Deepseek 672B
7B	Mathstral [19] A small model released by Mistral AI in 2024	DeepSeek 7B [6] A Qwen distilled version of Deepseek 672B
14B	Phi4 [1] A model released by Microsoft in 2024	DeepSeek 14B [6] A Qwen distilled version of Deepseek 672B
70B	Llama3.3 70B [3] A model released by MetaAI in 2024	DeepSeek 70B [6] A Llama distilled version of Deepseek 672B
400B+	Llama3.1 405B [8] A large model released by MetaAI in 2024	DeepSeek 671B [5] A large model released by DeepSeek AI in January 2025

B Additional Algorithms

This section contains the helper functions for Algorithms 3.1 and 3.2. Algorithm B.1 (Attempt_parse) returns True and None if no formulas in a list contain syntax errors. If there are error messages it returns false and the error messages.

Algorithm B.1: Attempt_parse: Checks and Parses LLM Equations

Input: *equation_list*
Output: *correct_syntax* (boolean), *error_messages* (if applicable)

```
try
    Parse equation_list through system;
    return True, None;
catch
    ParsingError;
    return False, error_messages;
```

Algorithm B.2 checks whether two lists of student equations are equivalent using the process described for equivalence in 3.2.

Algorithm B.2: Check_Equivalence: of equation Lists

Input: *student_equations*, *llm_equations*
Output: **True** if lists are equivalent; otherwise **False**

```
foreach eq_student in student_equations do
    found ← False;
    foreach eq_llm in llm_equations do
        if Z3Entailment(eq_student, eq_llm) then
            found ← True;
            break;
    if found = False then
        return False;
foreach eq_llm in llm_equations do
    found ← False;
    foreach eq_student in student_equations do
        if Z3Entailment(eq_llm, eq_student) then
            found ← True;
            break;
    if found = False then
        return False;
return True;
```

Algorithm B.3 describes the `Z3Entailment` function used in Algorithm B.2. This function calls an SMT solver to check if two equations can be shown to be unsatisfiable.

Algorithm B.3: Z3Entailment: Check Equation Entailment using Z3

Input: eq_1, eq_2
Output: **True** if eq_1 entails eq_2; otherwise **False**

try
  $eq1\_z3 \leftarrow ParseToZ3(eq_1)$;
  $eq2\_z3 \leftarrow ParseToZ3(eq_2)$;
catch
  ParsingError;
  return ***False***;
$prog \leftarrow InitializeZ3Program()$;
`/* Refutational formulation of checking whether eq1 entails eq2: */`
Add constraint $eq1\_z3$ to $prog$;
Add constraint $\neg(eq2\_z3)$ to $prog$;
$result \leftarrow ExecuteZ3(prog)$;
if $result = UNSAT$ **then**
  return ***True***;
else
  return ***False***;

C Extra Data Tables

Once the initial experiments were conducted, additional targeted experiments were performed with lower quantisations of models with strong performance just above the reasonable time requirement. The results of these experiments are shown in Table 4. Llama 3.3 70B and Deepseek-r1 70B were able to be run on a local machine using 4-bit floating point numbers (quantisation).

Table 4. Raw data for the additional experiments.

Model	Llama 3.3	Deepseek	Phi4	Phi4
Model Size (B)	70	70	14	14
Quantisation (bit)	4	4	8	4
Direct				
Time (s)	6,763	89,566	624	314
Tokens	5,268	89,871	5,471	5,497
Correct Answers	134	133	142	141
LLM-Modulo SymPy				
Time (s)	6,978	90,625	651	337
Tokens	5,461	91,003	5,974	5,732
Correct Answers	134	133	142	141
After Consensus	144	148	141	141
LLM-Modulo Z3				
Time (s)	7,183	126,033	796	377
Tokens	5,537	99,298	7,104	6,785
Correct Answers	141	146	147	144