

Article

Model Checking Using Large Language Models—Evaluation and Future Directions

Sotiris Batsakis ^{1,2,*}, Ilias Tachmazidis ^{2,†}, Matthew Mantle ^{2,‡}, Nikolaos Papadakis ^{1,‡} and Grigoris Antoniou ^{3,*}

¹ Electrical and Computer Engineering Department, Hellenic Mediterranean University, 71004 Heraklion, Greece; npapadak@hmu.gr

² School of Computing and Engineering, University of Huddersfield, Queensgate, Huddersfield HD1 3DH, UK; i.tachmazidis@hud.ac.uk (I.T.); m.e.mantle@hud.ac.uk (M.M.)

³ School of Built Environment, Engineering and Computing, Leeds Beckett University, Leeds LS1 3HE, UK

* Correspondence: sbatsakis@hmu.gr (S.B.); g.antoniou@leedsbeckett.ac.uk (G.A.)

† Current address: Department of Electrical and Computer Engineering, Hellenic Mediterranean University, 71410 Heraklion, Greece.

‡ These authors contributed equally to this work.

Abstract: Large language models (LLMs) such as ChatGPT have risen in prominence recently, leading to the need to analyze their strengths and limitations for various tasks. The objective of this work was to evaluate the performance of large language models for model checking, which is used extensively in various critical tasks such as software and hardware verification. A set of problems were proposed as a benchmark in this work and three LLMs (GPT-4, Claude, and Gemini) were evaluated with respect to their ability to solve these problems. The evaluation was conducted by comparing the responses of the three LLMs with the gold standard provided by model checking tools. The results illustrate the limitations of LLMs in these tasks, identifying directions for future research. Specifically, the best overall performance (ratio of problems solved correctly) was 60%, indicating a high probability of reasoning errors by the LLMs, especially when dealing with more complex scenarios requiring many reasoning steps, and the LLMs typically performed better when generating scripts for solving the problems rather than solving them directly.

Keywords: model checking; large language models; non-monotonic reasoning



Academic Editors: Vangelis Marinakis, George A. Tsihrintzis, Nikolaos G. Bourbakis and Maria Virvou

Received: 14 December 2024

Revised: 16 January 2025

Accepted: 17 January 2025

Published: 20 January 2025

Citation: Batsakis, S.; Tachmazidis, I.; Mantle, M.; Papadakis, N.; Antoniou, G. Model Checking Using Large Language Models—Evaluation and Future Directions. *Electronics* **2025**, *14*, 401. <https://doi.org/10.3390/electronics14020401>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Large language models (LLMs) have recently gained in prominence due to their exceptional performance in various language-related tasks, and they are the underlying technology behind chat bots such as ChatGPT (Available at: <https://chat.openai.com/> (accessed on 1 December 2024)). LLMs such as LaMDA [1] and GPT, which powers ChatGPT (with GPT-4 [2] being the version used as of November 2024 at ChatGPT), are based on training deep neural networks with billions of parameters over huge lexical datasets, often employing human judgment in a semi-supervised (for example, reinforcement learning) training setting [3,4]. The exceptional human-level performance of LLMs in several tasks has led to a widespread discussion about the potential benefits and dangers of such technologies in various areas of human society in general, including petitions to pause research on more capable LLMs [5]. For example GPT-4 achieved human-level performance in various academic and professional exams, including a score in the top 10% of test takers in the Uniform Bar Examination, and this performance has been attributed, to a large degree, to scaling LLMs to larger training datasets and more complex models with a larger number of parameters [2].

The impressive performance of LLMs, including their ability to demonstrate emerging intelligent behavior and reasoning capabilities, lead to the point of considering them forerunners of artificial general intelligence [6]. However, several issues related to LLMs have been identified, including the energy cost of training [7,8], the difficulty in controlling their behavior [9], ensuring conformity with stakeholder requirements and norms, and interpreting their functionality [10]. The interpretability of LLMs is a crucial issue, since neural-network-based LLMs appear to be ‘black boxes’, in contrast to logic based systems, and although various attempts have tried to deal with this problem, including the use of LLMs to interpret LLMs [11], this is still an unresolved issue. In addition, since LLMs are based on vast amounts of raw text, they tend to replicate their input rather than apply robust reasoning [12]. The use of raw text, instead of structured knowledge bases integrating machine-readable semantics, contributes to the difficulty in achieving efficient reasoning, and this is an issue examined in various works, such as [13–15] and surveyed in [16], with GSM-symbolic [17] and FrontierMath [18] being recent works on this topic. Generic surveys on LLM evaluation, including reasoning capabilities, were presented in [19,20], while the performance of LLMs in strategic reasoning was included in [21]. LLMs have also been used for testing [22], with mixed results, and in [23], a self-consistency strategy (i.e., sampling from different reasoning paths) was proposed for improving the reasoning process of LLMs. Using deep neural networks for knowledge graph reasoning was presented in [24,25]. Various attempts at integrating knowledge graphs (KGs) into LLMs have been proposed [26,27] as a solution to efficient reasoning. However, recent advances in LLM capabilities include high performance in the abovementioned academic and professional exams [2]. Moreover, the intense competition between major LLMs drives rapid improvements in the performance achieved by LLMs in knowledge representation and reasoning tasks (for example, in the PaLM 2 technical report available at: <https://ai.google/static/documents/palm2techreport.pdf> (accessed on 1 December 2024) and GPT-4 technical report at: <https://cdn.openai.com/papers/gpt-4.pdf> (accessed on 1 December 2024)). Recent advances in LLM capabilities have illustrated the need for an updated evaluation of the reasoning capabilities of LLM-based systems. This updated evaluation should take into account recent developments in the field, including the deployment of systems such as ChatGPT, Claude, and Gemini, employing the benefits of scalability [28] as well as the LLMs’ demonstrated ability to adjust to new tasks given just a small number of examples [29]. Furthermore, LLMs’ capabilities with respect to important methods such as model checking [30] have not yet been examined.

This work is a first step towards analyzing the performance, capabilities and limitations of LLMs for model checking, which can be defined as a set of methods for evaluating the behavior of finite-state systems and their properties [30]. Model checking has been used extensively in various domains such as hardware and software design, which includes the specification of liveness and safety properties typically expressed using temporal logics such as linear temporal logic (LTL) and computation tree logic (CTL) [31]. Model checking can be also used to solve constraint satisfaction problems and various problems that can be reduced to them. In this work, several problems that can be solved using model checking were solved using ChatGPT, which is powered by the state-of-the-art GPT-4 LLM [32], Claude powered by Claude 3.5 Sonnet (available at: <https://claude.ai/> (accessed on 1 December 2024)), and Google Gemini (Available at: <https://gemini.google.com/app> (accessed on 1 December 2024)) introduced in [33]. The responses of ChatGPT, Claude, and Gemini were then evaluated with respect to the ground truth by employing specialized model checking tools. In summary, the contribution of the current work is the following: (a) proposes a set of problems as a benchmark for evaluating the model checking capabilities of LLMs, (b) evaluates the performance of the current state-of-the-art LLMs for model

checking, and (c) critically analyzes the performance of LLMs for model checking and proposes future directions of related research.

2. Methodology

The performance of ChatGPT, Claude, and Gemini was evaluated using fifteen diverse problems that can be solved using model checking. The problems were given as input to ChatGPT, Claude, and Gemini, and the responses were compared to the ground truth. All problems were solved using specialized model checking tools such as Pro B [34], the correct solutions were manually verified by the authors, and the correct solutions to the problems are presented along with the corresponding responses of the LLMs. The fifteen evaluation problems were as follows:

Problem 1. *A simple Boolean formula in conjunctive normal form (CNF), i.e., a conjunction of clauses, was given as input to ChatGPT, Claude, and Gemini, and the LLMs were asked to check if the formula was satisfiable (i.e., an instance of a satisfiability -SAT- problem) and if yes to provide a solution. The formula is $(x1 \vee x2 \vee x5 \vee x4) \wedge (x1 \vee x2 \vee \neg x5 \vee x4) \wedge (x3 \vee x6) \wedge (\neg x4 \vee x7 \vee x1) \wedge (\neg x4 \vee \neg x7 \vee x2)$.*

Problem 2. *The following Logic puzzle from [34,35] was given to ChatGPT, Claude, and Gemini:*
Knights: always tell the truth.
Knaves: always lie.
1: A says: "B is a knave or C is a knave".
2: B says "A is a knight".
What are A & B & C?

Problem 3. *"(A) Input: one 3 gallon and one 5 gallon jug, and we need to measure precisely 4 gallons. (B) Having one 3 gallon and one 9 gallon jug, we need to measure precisely 7 gallons". Notice that, while Problem 3A is solvable, Problem 3B is not, since the greatest common divisor of the size of the two jugs must be a divisor of the quantity to measure [36].*

Problem 4. *This problem is an example of ProB [34] system distribution: Find a different digit (between 0 and 9) for each capital letter in the following equation:*

$$KISS * KISS = PASSION$$

Problem 5. *Description: "Someone in Dreadsbury Mansion killed Aunt Agatha. Agatha, the butler, and Charles live in Dreadsbury Mansion, and are the only ones to live there. A killer always hates, and is no richer than his victim. Charles hates no one that Agatha hates. Agatha hates everybody except the butler. The butler hates everyone not richer than Aunt Agatha. The butler hates everyone whom Agatha hates. No one hates everyone. Who killed Agatha?" This is a standard benchmark for theorem proving introduced in [37].*

Problem 6. *Description "A bank van had several bags of coins, each containing either 16, 17, 23, 24, 39, or 40 coins. While the van was parked on the street, thieves stole some bags. A total of 100 coins were lost. It is required to find how many bags were stolen. You may assume, if needed, that there are multiple bags for each number of coins." This is an instance of the Knapsack problem from [38] with an example solution distributed with the ProB model checking tool.*

Problem 7. *Description (from ProB system documentation, examples section): Assign the numbers 1...8 to vertices A...H in Figure 1 such that the values of the connected vertices differ by more than one.*

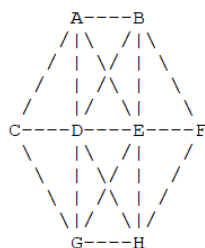


Figure 1. Graph of problem 7 (Source: ProB model checking system examples).

Problem 8. Can you place two queens and seven knights on a 6×6 chess board? (This is an example problem from the ProB system distribution).

Problem 9. The following NuSMV program from [39] is an example of an asynchronous model. It uses a variable semaphore to implement mutual exclusion between two asynchronous processes. Each process has four states: idle, entering, critical, and exiting. The entering state indicates that the process wants to enter its critical region. If the variable semaphore is FALSE, it goes to the critical state, and sets semaphore to TRUE. On exiting its critical region, the process sets semaphore to FALSE again.

```

MODULE main
  VAR
    semaphore : boolean;
    proc1 : process user(semaphore);
    proc2 : process user(semaphore);
  ASSIGN
    init(semaphore) := FALSE;
  MODULE user(semaphore)
    VAR
      state :
        {idle, entering, critical, exiting};
    ASSIGN
      init(state) := idle;
      next(state) :=
        case
          state = idle :
            {idle, entering};
          state = entering
            & !semaphore : critical;
          state = critical :
            {critical, exiting};
          state = exiting : idle;
          TRUE : state;
        esac;
      next(semaphore) :=
        case
          state = entering : TRUE;
          state = exiting : FALSE;
          TRUE : semaphore;
        esac;
  FAIRNESS
  Running

```

A desired property for this program is that it should never be the case that “the two processes *proc1* and *proc2* are at the same time in the critical state” (this is an example of a “safety” property). (A) Can you check if the property holds? (B) Can you express the property in CTL temporal logic?

Another desired property is that “if *proc1* wants to enter its critical state, it eventually does” (this is an example of a “liveness” property). (C) Can you check if the property holds? (D) Can you express the property in CTL temporal logic?

Problem 10. The input is the same problem as Problem 9 from [39] but properties must be represented using LTL temporal logic. Specifically the question is “Repeat for the same program as before but using LTL for checking the following properties: (A) Can you express the specification that the two processes cannot be in the critical region at the same time using LTL temporal logic and check if the property holds? (B) Can you express that whenever a process wants to enter its critical session, it eventually does using LTL and check if the property holds?”

Problem 11. “We try to place as many bishops as possible on a 8 by 8 chess board. Can you find the maximum number of bishops and their positions on the chess board?” This is an optimization problem (maximizing the number of bishops) under a specific hard constraint on the positions of bishops.

Problem 12. “We have the following information:

1. There are three boxes, one contains only pencils, one contains only pens, and one contains both pencils and pens.
2. The boxes have been incorrectly labeled such that no label identifies the actual contents of the box it labels.
3. Opening just one box, and without looking in the box, you take out one object.

By looking at the object, how can you immediately label all of the boxes correctly?” This example problem in ProB distribution is a variant of an Apple job interview puzzle.

Problem 13. “There is a table with room for three boxes. There are three boxes, a Red, a Green and a Blue box. The Red box is on the table. The Blue box is on the Red and the Green box is on the table. (A) Can you move boxes one by one so as to have the Red box on the Green, the Green on Blue, and the Blue one on the table? (B) Can you also solve the same problem when there is room for two boxes on the table?” These instances are examples of a blocks world, which is a standard benchmark for planning because of its complexity [40]. The first problem instance is solvable, while the second is not.

Problem 14. “Given the following Sudoku grid can you provide a solution?” Grid:

```
{ {3, 0, 6, 5, 0, 8, 4, 0, 0},
  {5, 2, 0, 0, 0, 0, 0, 0, 0},
  {0, 8, 7, 0, 0, 0, 0, 3, 1},
  {0, 0, 3, 0, 1, 0, 0, 8, 0},
  {9, 0, 0, 8, 6, 3, 0, 0, 5},
  {0, 5, 0, 0, 9, 0, 6, 0, 0},
  {1, 3, 0, 0, 0, 0, 2, 5, 0},
  {0, 0, 0, 0, 0, 0, 0, 7, 4},
  {0, 0, 5, 2, 0, 6, 3, 0, 0} }
```

Problem 15. “There are four people: Roberta, Thelma, Steve, and Pete. Among them, they hold eight different jobs. Each holds exactly two jobs. The jobs are: chef, guard, nurse, clerk, police officer (gender not implied), teacher, actor, and boxer. The job of nurse is held by a male. The husband of the chef is the clerk. Roberta is not a boxer. Pete has no education past the ninth grade. Roberta, the chef,

and the police officer went golfing together. Who holds which job?" This problem called the jobs problem was proposed in [41] and example solutions were proposed in [42] and further analyzed in [43].

The 15 Problems given to ChatGPT, Claude, and Gemini are all example problems solvable using model checking tools such as ProB [34] and NuSMV [44] (or the newest NuXMV tool [45]) and the answers of the evaluated LLMs were compared to the ground truth provided by the model checking tools. Solutions to Problems 2–8, 11–15 are available as model checking examples in the ProB model checking tool distribution (Available at: <https://prob.hhu.de/> (accessed on 1 December 2024)). Solutions to Problems 9 and 10 were verified using the NuSMV model checking tool and they are available as model checking examples in the NuSMV distribution (Available at: <https://nusmv.fbk.eu/> (accessed on 1 December 2024)).

3. Evaluation Results

During the evaluation process, the three tested LLMs (ChatGPT based on GPT-4, Claude, and Gemini) were prompted to solve the 15 benchmark problems. Their responses were compared to the ground truth obtained using model checking tools such as ProB, NuSMV, or manually. The results are presented in the following. Due to space limitations, most responses are presented partially, highlighting the main points, although a few example responses are fully listed.

3.1. Model Checking Using ChatGPT

The responses of ChatGPT to all problems were compared to the ground truth and the results were the following (the ChatGPT session is available at: <https://chatgpt.com/share/674cdee2-ba3c-800e-86cb-b55cfda52bb8> (accessed on 1 December 2024)):

Problem 16. ChatGPT Response: *To determine if a given Boolean formula is satisfiable, we can use a SAT solver, which checks whether there exists a combination of Boolean values (true/false) for the variables that satisfies all the given constraints. Your formula in conjunctive normal form (CNF) is: $(x_1 \vee x_2 \vee x_5 \vee x_4) \wedge (x_1 \vee x_2 \vee \neg x_5 \vee x_4) \wedge (x_3 \vee x_6) \wedge (\neg x_4 \vee x_7 \vee x_1) \wedge (\neg x_4 \vee \neg x_7 \vee x_2)$*

I'll implement a solver to check its satisfiability using Python with a SAT-solving library like pysat. Let me provide the solution. It seems that the required SAT-solving library, pysat, is not available in this environment. However, you can run the following Python code on your local machine to determine satisfiability:

```
from pysat.solvers import Glucose3

# Initialize the SAT solver
solver = Glucose3()

# Define the clauses in CNF form
# Each variable is represented as an integer.
# Positive integers represent variables,
# and negative integers represent their negations.

clauses = [
    [1, 2, 5, 4],          #  $x_1 \vee x_2 \vee x_5 \vee x_4$ 
    [1, 2, -5, 4],        #  $x_1 \vee x_2 \vee \neg x_5 \vee x_4$ 
    [3, 6],               #  $x_3 \vee x_6$ 
    [-4, 7, 1],          #  $\neg x_4 \vee x_7 \vee x_1$ 
]
```

```

    [-4, -7, 2]          # ¬x4 ∨ ¬x7 ∨ x2
]

# Add clauses to the solver
for clause in clauses:
    solver.add_clause(clause)

# Check for satisfiability
is_satisfiable = solver.solve()

# Get a satisfying assignment if it exists
assignment = solver.get_model() if is_satisfiable else None

# Output results
print("Is the formula satisfiable?", is_satisfiable)
if is_satisfiable:
    print("Satisfying assignment:", assignment)

# Clean up the solver
solver.delete()

```

Steps to Run Locally Install the pysat library: pip install python-sat. Run the script above in any Python environment. The output will indicate whether the formula is satisfiable, and if it is, it will provide a satisfying assignment.

Evaluation: *The output of the program is "Is the formula satisfiable? True Satisfying assignment: [1, -2, 3, -4, -5, -6, -7]".*

The solution is correct, demonstrating the basic reasoning capabilities of ChatGPT/GPT-4.

Problem 17. ChatGPT Response: *"To solve this problem, we analyze the statements logically based on the rules: 1. Knights always tell the truth. 2. Knaves always lie... Step-by-Step Analysis Case 1: A is a Knight... From the analysis: A is a knight. B is a knave. C is a knight. This satisfies all the conditions and the truth/falsehood of the statements".*

Evaluation: *Manual inspection of the problem and also formal model checking using ProB returned the solution A and B are knights and C is a knave. ChatGPT generated sentences that seem correct but the reasoning process is actually faulty, thus failing to solve this problem.*

Problem 18. ChatGPT Response: *"To solve these problems, we can use the water jug problem strategy, which involves transferring water between jugs to achieve the desired measurements. Let's tackle each problem step by step. Problem A: 3-Gallon and 5-Gallon Jugs, Measure 4 Gallons We denote the amount of water in the 3-gallon jug as xxx and in the 5-gallon jug as yyy. The steps are:*

1. *Fill the 5-gallon jug completely: $(x,y) = (0,5)$.*
2. *Pour water from the 5-gallon jug into the 3-gallon jug until the 3-gallon jug is full: $(x,y) = (3,2)$.*
3. *Empty the 3-gallon jug: $(x,y) = (0,2)$.*
4. *Pour the remaining water from the 5-gallon jug into the 3-gallon jug: $(x,y) = (2,0)$.*
5. *Fill the 5-gallon jug completely again: $(x,y) = (2,5)$.*
6. *Pour water from the 5-gallon jug into the 3-gallon jug until the 3-gallon jug is full: $(x,y) = (3,4)$.*

Now, the 5-gallon jug contains exactly 4 gallons of water. Problem B: 3-Gallon and 9-Gallon Jugs, Measure 7 Gallons We denote the amount of water in the 3-gallon jug as xxx and in the 9-gallon jug as yyy. The steps are:

1. Fill the 9-gallon jug completely: $(x,y) = (0,9)$.
2. Pour water from the 9-gallon jug into the 3-gallon jug until the 3-gallon jug is full: $(x,y) = (3,6)$.
3. Empty the 3-gallon jug: $(x,y) = (0,6)$.
4. Pour water from the 9-gallon jug into the 3-gallon jug until the 3-gallon jug is full: $(x,y) = (3,3)$.
5. Empty the 3-gallon jug: $(x,y) = (0,3)$.
6. Pour the remaining water from the 9-gallon jug into the 3-gallon jug: $(x,y) = (3,0)$.
7. Fill the 9-gallon jug completely again: $(x,y) = (3,9)$.
8. Pour water from the 9-gallon jug into the 3-gallon jug until the 3-gallon jug is full: $(x,y) = (3,7)$.

Now, the 9-gallon jug contains exactly 7 gallons of water.

Summary of Results: “Problem A: The 5-gallon jug contains 4 gallons after following the steps. Problem B: The 9-gallon jug contains 7 gallons after following the steps”.

Evaluation: The response on part A is correct, but the response on part B is not correct. The instance in part B is not solvable, and there is an error at step 8 of the response.

Problem 19. ChatGPT Response: “This is a cryptarithm, where each letter represents a unique digit, and we need to find values such that the given equation is satisfied: $KISS \times KISS = PASSION$... Let me refine the search or confirm that no solution exists. I’ll double-check for possible edge cases and ensure the logic is correct... (python program) ...

It appears that no valid solutions exist for the cryptarithm $KISS \times KISS = PASSION$ under the given constraints. This could imply... Otherwise, we can conclude that the equation has no solution with unique digits”.

Evaluation: The ProB model checking tool provided the answer: $2033 \times 2033 = 4,133,089$, so ChatGPT failed to provide the correct answer, since when running the suggested code no solutions were returned.

Problem 20. ChatGPT Response: “Logical Breakdown. 1. Key Facts:... Step-by-Step Analysis Step 1: Could Charles have killed Agatha?... Therefore, Charles cannot be the killer.... Step 2: Could the butler have killed Agatha?... The butler satisfies all the conditions to be the killer. Step 3: Could Agatha have killed herself?... There is no indication that Agatha hates herself, so she cannot be the killer. Conclusion The butler killed Aunt Agatha”.

Evaluation: The solution of the problem in [37] and also the verification using ProB model checking tool is that Agatha killed herself. Although ChatGPT provided a sequence of assertions that seemed logical overall, the reasoning process was false.

Problem 21. ChatGPT Response: “To solve this problem, we need to determine how many bags were stolen such that the total number of coins lost is 100. Each bag contains... This is a Diophantine equation problem, where we need to find non-negative integer solutions to: $16x_1 + 17x_2 + 23x_3 + 24x_4 + 39x_5 + 40x_6 = 100$... Approach 1. Solve the equation programmatically using an exhaustive search. 2. Minimize the total number of bags stolen... However, you can solve this problem using the following Python code on your local machine:...

This code will find the combinations of bags that result in exactly 100 coins and minimize the total number of bags stolen.”

Evaluation: Running the code returned the correct result, [16, 16, 17, 17, 17, 17], which was verified using ProB model checking tool, which also provided the answer: $2 \times 16 + 4 \times 17$.

Problem 22. ChatGPT Response: This problem involves assigning the integers 1, 2, ..., 8... Graph Representation The graph has the following edges:... We will: 1. Represent the graph as a set of edges...

Evaluation: Running the code returned A: 3, B: 5, C: 7, D: 1, E: 8, F: 2, G: 4, H: 6. The ProB model checker verified the answer, which demonstrated the ability of ChatGPT to generate source code that solved the problem.

Problem 23. ChatGPT Response: To solve the problem of placing two queens and seven knights on a 6×6 . . . Queens: Attack all squares in the same row, column, and diagonal. Knights: Attack in an "L" shape (two squares in one direction and one square perpendicular). . . Approach 1. Place Queens: Ensure the two queens do not attack each other. They cannot share the same row, column, or diagonal. 2. Place Knights: . . .

Q	0	0	0	0	0
0	0	K	Q	0	0
0	K	0	0	0	K
0	0	K	0	K	0
0	K	0	0	0	K
0	0	0	0	0	0

where: Q represents a queen, . . . This placement ensures no piece attacks another."

Evaluation: The response of ChatGPT to the problem (and the output of the python program in the response) was incorrect. The ProB solver provided the solution to the problem, as shown in Figure 2.

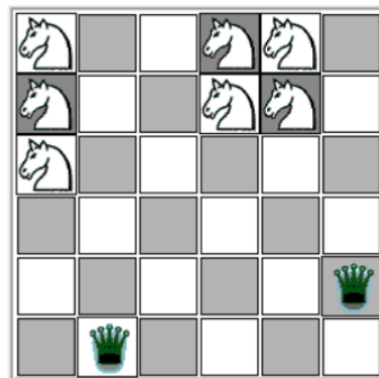


Figure 2. Solution of problem 8.

Problem 24. ChatGPT Response: "(A) Can the safety property hold? The safety property is that proc1 and proc2 should never simultaneously be in the critical state. We analyze whether the program enforces mutual exclusion using the semaphore variable. . . Since semaphore = FALSE is a necessary condition for entering the critical state, and a process sets semaphore = TRUE immediately upon entering critical, it is impossible for both proc1 and proc2 to be in the critical state simultaneously. . . (B) CTL Expression for the Safety Property. The property that "proc1 and proc2 are not simultaneously in the critical state" can be expressed in CTL (Computation Tree Logic) as:

$$AG \neg (proc1.state = critical \wedge proc2.state = critical)$$

This asserts that in all states (A) and along all paths (G), it is never the case (\neg) that both processes are in the critical state simultaneously. (C) Can the liveness property hold? The liveness property states that if proc1 wants to enter the critical state (i.e., is in the entering state), it will eventually reach the critical state. . . . There is no mechanism to prioritize proc1 over proc2. If proc2 repeatedly enters the critical state immediately after exiting, proc1 may be indefinitely delayed (starvation). Conclusion: The liveness property does not hold, as there is no guarantee that proc1 will eventually enter the critical state due to potential starvation. (D) CTL Expression for the Liveness Property The liveness property, if it were to hold, can be expressed in CTL as: $AG(proc1.state = entering \rightarrow AF$

This asserts that whenever (A) *proc1* is in the entering state (G), it is always the case (A) that it will eventually (F) reach the critical state.

Summary

Safety Property: Holds.

CTL: $\mathbf{AG}\neg(\text{proc1.state} = \text{critical} \wedge \text{proc2.state} = \text{critical})$

Liveness Property: Does not hold (starvation is possible).

CTL: $\mathbf{AG}(\text{proc1.state} = \text{entering} \rightarrow \mathbf{AF}(\text{proc1.state} = \text{critical}))$

Evaluation: The response and the translation to CTL was correct. The translation, when used with the NuSVM model checking tool to check the properties, provided the correct answer (i.e., the first property holds, while the second does not).

Problem 25. ChatGPT Response: (A) Safety Property: Processes cannot be in the critical region at the same time. LTL Expression: The safety property that *proc1* and *proc2* cannot simultaneously be in the critical state can be expressed in LTL (Linear Temporal Logic) as:

$$\mathbf{G}\neg(\text{proc1.state} = \text{critical} \wedge \text{proc2.state} = \text{critical})$$

This reads: “Globally (G), it is always the case that both processes are not (\neg) in the critical state at the same time.” Does the Property Hold? The program uses the variable semaphore to enforce mutual exclusion: . . . Since mutual exclusion is correctly implemented, the safety property holds under the given program. (B) Liveness Property: Whenever a process wants to enter its critical section, it eventually does. LTL Expression: The liveness property for a process (e.g., *proc1*) can be expressed in LTL as:

$$\mathbf{G}(\text{proc1.state} = \text{entering} \rightarrow \mathbf{F}(\text{proc1.state} = \text{critical}))$$

This reads: “Globally (G), if *proc1* is in the entering state, then it is eventually (F) true that *proc1* reaches the critical state.” Similarly, for *proc2*, the property would be:

$$\mathbf{G}(\text{proc2.state} = \text{entering} \rightarrow \mathbf{F}(\text{proc2.state} = \text{critical}))$$

Does the Property Hold? The liveness property depends on the program’s fairness and the possibility of starvation: . . . As a result, the liveness property does not hold in general, because starvation is possible.

Summary of Results

1. Safety Property:

LTL: $\mathbf{G}\neg(\text{proc1.state} = \text{critical} \wedge \text{proc2.state} = \text{critical})$

Holds: The program ensures mutual exclusion.

2. Liveness Property: LTL: $\mathbf{G}(\text{proc1.state} = \text{entering} \rightarrow \mathbf{F}(\text{proc1.state} = \text{critical}))$

Does Not Hold: Starvation is possible because there is no prioritization mechanism.”

Evaluation: Both properties were correctly translated to LTL, as verified using the NuSMV model checking tool.

Problem 26. ChatGPT Response: “The maximum number of bishops that can be placed on an 8×8 chessboard is determined by the nature of their movements: Bishops attack diagonally, so two bishops can only share the same diagonal if they are on different colors. . . unique diagonals for bishops (considering both directions). A bishop placed on one diagonal prevents any other bishop from being placed on the same diagonal. Result: The maximum number of bishops that can be placed is $n = 8$. This is because there are 8 rows and we can ensure that each row and diagonal contains at most one bishop. Placement: We need to assign one bishop to each row while ensuring no two bishops share the same diagonal. . .

(python code)...

Evaluation: The solution of ChatGPT was incorrect. The correct solution with 14 bishops, found using ProB, is shown in Figure 3.

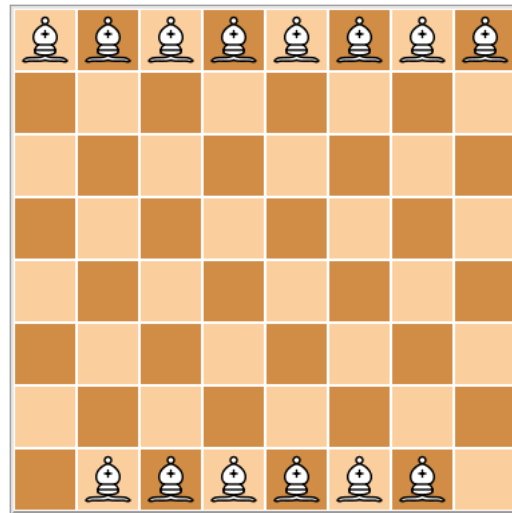


Figure 3. Solution of problem 11.

Problem 27. ChatGPT Response: “To solve this problem, consider the labels and their incorrectness, and use the given conditions strategically:

Step 1: Key Insight: Since all the boxes are incorrectly labeled: The box labeled “Pencils and Pens” cannot contain both pencils and pens. It must contain only pencils or only pens. . .

Step 2: Open the Box Labeled “Pencils and Pens” 1. Take one object from the box labeled “Pencils and Pens.” If you pull out a pencil, this box must contain only pencils (because it cannot contain both pencils and pens due to the incorrect label). If you pull out a pen, this box must contain only pens.

Step 3: Deduce the Other Boxes: After determining the contents of the box labeled “Pencils and Pens”, you can label it correctly as either “Pencils” or “Pens.” The remaining two boxes are now straightforward: The box originally labeled “Pencils” must contain the other single-object type. The box originally labeled “Pens” must contain both pencils and pens.

Example Walkthrough

Case 1: You draw a pencil from the box labeled “Pencils and Pens”: 1. This box must contain only pencils. 2. The box labeled “Pencils” (incorrect label) must contain both pencils and pens. 3. The box labeled “Pens” (incorrect label) must contain only pens.

Case 2: You draw a pen from the box labeled “Pencils and Pens”: 1. This box must contain only pens. 2. The box labeled “Pencils” (incorrect label) must contain both pencils and pens. 3. The box labeled “Pens” (incorrect label) must contain only pencils.

Evaluation: The response was mostly correct, including both the first step and description of the deduction process, but there was an error in the example walkthrough (In case 1).

Problem 28. ChatGPT Response: “Problem Analysis: We need to rearrange the boxes (Red, Green, Blue) from their initial state to a final state: Initial State: Red (R) is on the table. . . We solve this problem under two conditions: 1. (A) The table can hold three boxes. 2. (B) The table can hold only two boxes.

(A) Solution with Room for Three Boxes on the Table: Here, the table has room for three boxes, so we can freely move boxes back and forth without concern about exceeding the capacity.

Step-by-step Solution:

1. Move Blue (B) from Red (R) to the table.
2. Move Green (G) onto Blue (B).

3. Move Red (R) onto Green (G).

Final Arrangement: Blue (B) on the table. Green (G) on Blue (B). Red (R) on Green (G).

(B) Solution with Room for Two Boxes on the Table: Here, the table has room for only two boxes at a time. This restriction requires careful intermediate placements of boxes. Step-by-step Solution:

1. Move Blue (B) to the floor...

Summary of Moves

Case A: Room for Three Boxes

1. Move Blue to the table.

2. Move Green onto Blue.

3. Move Red onto Green.

Case B: Room for Two Boxes

1. Move Blue to the floor.

2. Move Green onto Red.

3. Move Blue onto the table.

4. Move Green onto Blue.

5. Move Red onto Green".

Evaluation: The solution of A was correct, but the solution of part B used actions that are not allowed (i.e., using the floor). Part B has no solution when only moving the object on the table is allowed.

Problem 29. ChatGPT Response:

3	1	6	5	7	8	4	9	2
5	2	9	1	3	4	7	6	8
4	8	7	6	2	9	5	3	1
2	6	3	4	1	5	9	8	7
9	7	4	8	6	3	1	2	5
8	5	1	7	9	2	6	4	3
1	3	8	9	4	7	2	5	6
6	9	2	3	5	1	8	7	4
7	4	5	2	8	6	3	1	9

Evaluation: The response was correct.

Problem 30. ChatGPT Response: "To solve this puzzle, we need to carefully analyze the clues and assign jobs to each person while adhering to the given constraints. Here's a step-by-step breakdown:

Clues: Each person holds exactly two jobs. Jobs: chef, guard, nurse, clerk, police officer, teacher, actor, boxer... Step 1: Assign constraints... Step 2: Narrow down roles for Pete and Steve... Step 3: Assign Roberta's second job... Step 4: Assign remaining jobs for Pete... Step 5: Assign jobs for Steve... Step 6: Assign remaining jobs for Thelma. Remaining jobs: actor and boxer. Since Roberta cannot be a boxer, Thelma must be the boxer and actor.

Final Assignments:

Roberta: Chef, Teacher.

Thelma: Actor, Boxer.

Steve: Police Officer, Guard.

Pete: Clerk, Nurse.

Evaluation: The response was not correct. The correct solution (provided by ProB) that satisfies all constraints is as follows: Thelma: chef, boxer, Roberta: guard, teacher, Steve: police officer, nurse, Pete: actor, clerk and Thelma is married to Pete.

ChatGPT Summary: Seven responses were correct, two were partially correct, and six were incorrect.

3.2. Model Checking Using Claude

The responses of Claude to all problems were compared to the ground truth and the results are shown in the following (full sessions of all LLMs with comments are available at: <https://github.com/sbatsakis/LLM-modelchecking.git> (accessed on 1 December 2024)).

Problem 31. Claude Response: *“I’ll help you solve this Boolean satisfiability problem. Since this is a complex satisfiability problem, I’ll write a Python script using the Z3 theorem prover, which is an excellent tool for solving such logical satisfiability problems.*

python:

```
from z3 import *

def solve_satisfiability():
    # Create Boolean variables
    x1 = Bool('x1')
    x2 = Bool('x2')
    x3 = Bool('x3')
    x4 = Bool('x4')
    x5 = Bool('x5')
    x6 = Bool('x6')
    x7 = Bool('x7')

    # Create the solver
    solver = Solver()

    # Add the constraints from the problem
    solver.add(Or(x1, x2, x5, x4))
    solver.add(Or(x1, x2, Not(x5), x4))
    solver.add(Or(x3, x6))
    solver.add(Or(Not(x4), x7, x1))
    solver.add(Or(Not(x4), Not(x7), x2))

    # Check satisfiability
    if solver.check() == sat:
        print("The formula is SATISFIABLE")
        # Get the model (one possible solution)
        model = solver.model()
        print("One possible assignment:")
        print(f"x1 = {model[x1]}")
        print(f"x2 = {model[x2]}")
        print(f"x3 = {model[x3]}")
        print(f"x4 = {model[x4]}")
        print(f"x5 = {model[x5]}")
        print(f"x6 = {model[x6]}")
        print(f"x7 = {model[x7]}")
    else:
        print("The formula is UNSATISFIABLE")
```

```
# Run the satisfiability check
solve_satisfiability()
```

To solve this Boolean satisfiability problem, I've used the Z3 theorem prover from Microsoft Research. Here's a breakdown of the approach:

1. We create Boolean variables x_1 through x_7 ...

To run this, you'll need to:...

Note: The Z3 solver will find one satisfying assignment if it exists, but there might be multiple possible solutions that satisfy the constraints."

Evaluation: After executing the code the result was as follows: The formula is SATISFIABLE. One possible assignment: $x_1 = \text{True}$, $x_2 = \text{False}$, $x_3 = \text{True}$, $x_4 = \text{False}$, $x_5 = \text{False}$, $x_6 = \text{False}$, $x_7 = \text{False}$, which was correct.

Problem 32. Claude Response: "I'll solve this logic puzzle using a systematic approach with a Python script to check all possible scenarios.

```
'''python
def is_valid_scenario(a_type, b_type, c_type):
    """
    Check if the given scenario is logically consistent
    a_type, b_type, c_type are either 'knight' or 'knave'
    """
    # A's statement: "B is a knave or C is a knave"
    a_statement = (b_type == 'knave') or (c_type == 'knave')

    \# B's statement: "A is a knight"
    b_statement = (a_type == 'knight')

    # Check if statements are consistent with character types
    if a_type == 'knight':
        if not a_statement:
            return False
    else: # a is a knave
        if a_statement:
            return False

    if b_type == 'knight':
        if not b_statement:
            return False
    else: # b is a knave
        if b_statement:
            return False

    return True

def solve_puzzle():
    types = ['knight', 'knave']

    # Try all possible combinations
    for a_type in types:
        for b_type in types:
```



```

        for c_type in types:
            if is_valid_scenario(a_type, b_type, c_type):
                print(f"Solution found:")
                print(f"A is a {a_type}")
                print(f"B is a {b_type}")
                print(f"C is a {c_type}")
                return

    print("No solution found")

# Solve the puzzle
solve_puzzle()

```

Let me break down the solution approach:...

When you run the script, it will output the first valid solution it finds that satisfies all the logical constraints.

The key is to carefully track the logical implications of each character's type and statement."

Evaluation: *The output of the python program was as follows: Solution found:*

A is a knight

B is a knight

C is a knave

The response was correct.

Problem 33. Claude Response: *"I'll solve these water jug measurement problems using Python. I'll create a systematic algorithm to find the minimum number of steps to measure the desired amount.*

```

'''python
def solve_water_jug_problem(jug1_capacity, jug2_capacity, target_amount):
    """
    Solve the water jug problem using breadth-first search

    Args:
    jug1_capacity (int): Capacity of the first jug
    jug2_capacity (int): Capacity of the second jug
    target_amount (int): The precise amount we want to measure

    Returns:
    list: Steps to measure the target amount, or None if impossible
    """
    # Track visited states to avoid infinite loops
    visited = set()

    # Queue to track states: (jug1, jug2, steps)
    queue = [(0, 0, [])]

    while queue:
        jug1, jug2, steps = queue.pop(0)

        # Check if we've reached the target
        if jug1 == target_amount or jug2 == target_amount:
            return steps

```

```

        # Avoid revisiting states
        if (jug1, jug2) in visited:
            continue
        visited.add((jug1, jug2))

        # Possible actions:
        # 1. Fill jug1
        queue.append((jug1_capacity, jug2, steps + ['Fill 3-gallon jug']))

        # 2. Fill jug2
        queue.append((jug1, jug2_capacity, steps + ['Fill 9-gallon jug']))

        # 3. Empty jug1
        queue.append((0, jug2, steps + ['Empty 3-gallon jug']))

        # 4. Empty jug2
        queue.append((jug1, 0, steps + ['Empty 9-gallon jug']))

        # 5. Pour from jug1 to jug2
        pour_to_jug2 = min(jug1, jug2_capacity - jug2)
        queue.append((
            jug1 - pour_to_jug2,
            jug2 + pour_to_jug2,
            steps + [f'Pour {pour_to_jug2} from 3-gallon to 9-gallon jug']
        ))

        # 6. Pour from jug2 to jug1
        pour_to_jug1 = min(jug2, jug1_capacity - jug1)
        queue.append((
            jug1 + pour_to_jug1,
            jug2 - pour_to_jug1,
            steps + [f'Pour {pour_to_jug1} from 9-gallon to 3-gallon jug']
        ))

    return None # No solution found

# Problem A: 3 and 5 gallon jugs, target 4 gallons
print("Problem A: 3 and 5 gallon jugs, target 4 gallons")
solution_a = solve_water_jug_problem(3, 5, 4)
if solution_a:
    print("Solution steps:")
    for step in solution_a:
        print(step)
else:
    print("No solution found")

print("\n" + "="*50 + "\n")

# Problem B: 3 and 9 gallon jugs, target 7 gallons
print("Problem B: 3 and 9 gallon jugs, target 7 gallons")

```

```

solution_b = solve_water_jug_problem(3, 9, 7)
if solution_b:
    print("Solution steps:")
    for step in solution_b:
        print(step)
else:
    print("No solution found")
'''

```

I've solved both water jug problems using a breadth-first search algorithm. Let me break down the key aspects:

1. *The algorithm explores all possible actions:*
 - *Filling a jug completely*
 - *Emptying a jug*
 - *Pouring water from one jug to another*
2. *For Problem A (3 and 5 gallon jugs, target 4 gallons): - The algorithm finds a solution by systematically trying different pour strategies*
 - *It will output the exact steps to measure 4 gallons*
3. *For Problem B (3 and 9 gallon jugs, target 7 gallons):*
 - *Similarly, it will find the minimum steps to measure 7 gallons*

The code uses a breadth-first search to:

- *Explore all possible states*
- *Avoid revisiting states to prevent infinite loops*
- *Find the shortest sequence of actions to reach the target*

Key advantages of this approach:

- *Guaranteed to find a solution if one exists*
- *Provides the exact steps to solve the problem*
- *Works for different jug sizes and target amounts*

Evaluation: *After running the code, the response was as follows: "Problem A: 3 and 5 gallon jugs, target 4 gallons Solution steps:*

*Fill 9-gallon jug
 Pour 3 from 9-gallon to 3-gallon jug
 Empty 3-gallon jug
 Pour 2 from 9-gallon to 3-gallon jug
 Fill 9-gallon jug
 Pour 1 from 9-gallon to 3-gallon jug
 Problem B: 3 and 9 gallon jugs, target 7 gallons
 No solution found"*

The response was correct for both cases.

Problem 34. Claude Response: *I'll solve this alphanumeric puzzle using a systematic approach with Python's constraint solving techniques... (python code)...*

Here's how the solution works:

1. *The script uses a brute-force approach to find a solution:...*
2. *Constraints enforced:...*
3. *The algorithm does several checks:...*

Solving strategy involves systematically exploring permutations until finding a valid solution that satisfies all constraints."

Evaluation: *After running the code, the output was as follows: "No solution found." The response was not correct.*

Problem 35. Claude Response: *“I’ll solve this logic puzzle by creating a systematic approach to analyze the constraints and determine the killer... (python code)...”*

Let me break down the solution approach:...

1. *The puzzle constraints are complex:...*
2. *The script systematically:*
 - *Tries different potential killers*
 - *Checks all possible hate relationships*
 - *Verifies each scenario against the given constraints*
3. *Key logical deductions:...*

When you run the script, it will output:

- *The killer*
- *The victim*
- *The hate relationships between characters*

The solution uses a constraint satisfaction approach to find the unique scenario that meets all the logical conditions in the murder mystery.”

Evaluation: *Running the code failed to return a solution. So problem 5 was not solved by Claude.*

Problem 36. Claude Response: *“I’ll solve this coin bag theft problem by creating a Python script that systematically finds the combination of bags that results in exactly 100 stolen coins... (python code)...”*

Here’s how the solution works:...

1. *Systematic Approach:*

- *The script tries all possible combinations of bags...*

2. *Key Constraints:*

- *Total stolen coins must be exactly 100... 3. Solution Strategy:... - Uses nested loops to explore*

all bag combination possibilities...

When you run the script, it will:

- *Show which bags were stolen (bag types and counts)...*

The brute-force approach ensures we find the exact solution that meets all the problem’s constraints.”

Evaluation: *The output of the program was*

“Solution found:

Bags stolen:

2 bag(s) with 16 coins

4 bag(s) with 17 coins

Total bags stolen: 6

Verification:

Total coins stolen: 100”

The response was correct.

Problem 37. Claude Response: *“I’ll solve this graph vertex numbering problem by creating a constraint satisfaction algorithm in Python... (python code)...”*

1. *Constraint Satisfaction Algorithm:*

- *Uses backtracking to explore possible number assignments...*

2. *Solution Strategy:...* 3. *Key Algorithm Components:...*

When you run the script, it will:...

The solution demonstrates how to solve a complex constraint satisfaction problem using a backtracking approach.”

Evaluation: *The output of the program was “Solution found: A: 3, B: 5, C: 7, D: 1, E: 8, F: 2, G: 4, H: 6.*

Verification: $A(3) - B(5)$: difference = 2... $H(6) - G(4)$: difference = 2. All connections satisfy the constraint!"

The solution was correct.

Problem 38. Claude Response: "I'll solve this chess piece placement problem using a constraint satisfaction approach... (python code)... Here's a breakdown of the solution approach: 1. Constraint Satisfaction Strategy:

- Use backtracking to systematically place pieces...

3. Solution Details:

- Uses a 6×6 board representation...

The solution demonstrates how to solve a complex piece placement problem using a constraint satisfaction and backtracking approach."

Evaluation: The output was "Solution found:

Q N N N N N

. . Q . . .

.

N . N . . .

.

.

Queens placed: 2 Knights placed: 7"

The response was incorrect, since there are attacks between Queens and Knights.

Problem 39. Claude Response: "I'll solve this concurrency and temporal logic problem using the NuSMV model checker, which is designed for verifying properties of concurrent systems

```

'''nusmv
-- NuSMV Model for Mutual Exclusion and Liveness Properties

MODULE main
VAR
    semaphore : boolean;
    proc1 : user(semaphore);
    proc2 : user(semaphore);

ASSIGN
    init(semaphore) := FALSE;

-- User process module
MODULE user(semaphore)
VAR
    state : {idle, entering, critical, exiting};

.....

FAIRNESS
    running

```

```

-- Safety Property A:
-- Verify that both processes are never in critical state simultaneously
SPEC
    AG!(proc1.state = critical & proc2.state = critical)

-- Liveness Property C:
-- If proc1 wants to enter critical state, it eventually does
SPEC
    AG(proc1.state = entering -> AF proc1.state = critical)

-- Additional Temporal Logic Expressions:

-- B) Safety Property in CTL:
-- AG(!(proc1.state = critical & proc2.state = critical))
-- This is the same as the SPEC above, expressed in pure CTL

-- D) Liveness Property in CTL:
-- AG((proc1.state = entering) -> AF(proc1.state = critical))
-- This is the same as the second SPEC above, expressed in pure CTL

'''

```

Let me break down the solution and explain the key aspects:

1. *Safety Property A (Mutual Exclusion): - The specification 'AG!(proc1.state = critical & proc2.state = critical)' means: * AG (Always Globally): Throughout all computation paths...*
2. *Liveness Property C (Progress): - The specification 'AG(proc1.state = entering -> AF proc1.state = critical)' means: * AG (Always Globally): Throughout all computation paths...*

3. *Temporal Logic Notation:*

- AG: Always Globally (for all future states)
- AF: Always Finally (eventually)
- !: Negation
- ->: Implies

4. *Model Characteristics:...*

To verify these properties, you would typically:

1. *Load this model in NuSMV*
2. *Run the model checking*
3. *Analyze the results*

Notes on the solutions:

- *Part B: The CTL expression is equivalent to the SPEC for safety*
- *Part D: The CTL expression is equivalent to the SPEC for liveness"*

Evaluation: *The solution for providing a translation to CTL was correct.*

Problem 40. Claude Response: *"I'll solve this using LTL (Linear Temporal Logic) temporal logic verification for the same concurrency model..."*

```

'''nusmv
-- NuSMV Model with LTL Properties for Mutual Exclusion and Progress

MODULE main
...

```



```

...
-- (A) LTL Property: Mutual Exclusion
-- Ensures the two processes are never simultaneously in critical state
LTLSPEC
    G!(proc1.state = critical & proc2.state = critical)

-- (B) LTL Property: Progress/Liveness
-- Ensures that whenever a process wants to enter critical section,
-- it will eventually enter
LTLSPEC
    G((proc1.state = entering) -> F(proc1.state = critical))

LTLSPEC
    G((proc2.state = entering) -> F(proc2.state = critical))

-- Additional Explanatory LTL Specifications

-- Variant of mutual exclusion showing explicit separation
LTLSPEC
    G(proc1.state = critical -> !proc2.state = critical)

-- Progress for both processes combined
LTLSPEC
    G(proc1.state = entering) -> F(proc1.state = critical)
    & G(proc2.state = entering) -> F(proc2.state = critical)

'''

```

Let me break down the LTL (Linear Temporal Logic) specifications:

1. Mutual Exclusion Property (A):
 - 'G!(proc1.state = critical & proc2.state = critical)'...
 2. Progress/Liveness Property (B):
 - 'G((proc1.state = entering) -> F(proc1.state = critical))'...
- To verify these properties, you would:...

The key difference from the previous CTL approach is the linear nature of LTL, which focuses on single computation paths rather than branching time."

Evaluation: The translation and tool suggestion were correct.

Problem 41. Claude Response: "I'll solve this problem for you using a mathematical approach and then provide a Python implementation to find the maximum number of bishops that can be placed on an 8x8 chess board.

The key insight for this problem is that bishops move diagonally, so to...

Here's a solution that demonstrates this:... (python code)...

Let me break down the solution:

1. Mathematical Insight:
 - Bishops can only move diagonally...
2. Solution Strategy:
 - On an 8×8 board, we can place bishops on diagonals without conflicts...
3. Implementation Details:
 - The code generates bishop positions on different diagonal types...

When you run this code, it will output:

- Maximum number of bishops (which is 14 for an 8×8 board)
 - The specific positions of these bishops on the board
- For an 8×8 board, the result is 14 bishops:
- 7 bishops on dark squares
 - 7 bishops on light squares"

Evaluation: Running the code yielded "Maximum number of bishops: 8 Bishop Positions (row, column): (0, 0) ... (2, 1)... (1, 1)... (1, 2)... (7, 6)" The number of bishops in the direct answer was correct, but the program output did not correspond to this number and the positions were not correctly specified, so the solution was incorrect.

Problem 42. Claude Response: "Let me solve this problem by walking you through the logical reasoning step by step.

The key to solving this problem is to leverage the given constraints that ALL labels are currently INCORRECT. Here's the solution:

1. Initial Situation:

- You have three boxes
- Current labels are WRONG
- Boxes contain:

- (a) Only pencils (b) Only pens
- (c) Both pencils and pens

2. Solution Strategy:

- Open the box labeled "Pencils and Pens"
- Since ALL labels are wrong, this box CANNOT contain both pencils and pens
- By taking out ONE object, you will know whether this box contains ONLY pencils or ONLY pens

3. Deduction Process: -If you pull out a PEN from the "Pencils and Pens" box, you know:

- * This box contains ONLY PENS
- * The box labeled "Only Pens" must actually contain PENCILS
- * The box labeled "Only Pencils" must contain BOTH pencils and pens
- If you pull out a PENCIL from the "Pencils and Pens" box, you know:
- * This box contains ONLY PENCILS
- * The box labeled "Only Pencils" must actually contain PENS
- * The box labeled "Only Pens" must contain BOTH pencils and pens

4. Final Labeling:

-By pulling out just one object from the "Pencils and Pens" box, you can definitively and correctly label ALL three boxes

The beauty of this solution is its elegant simplicity. By understanding that ALL current labels are incorrect, and by extracting a single object from one box, you gain enough information to correctly deduce the contents of all boxes."

Evaluation: The solution provided by Claude was correct.

Problem 43. Claude Response: "I'll solve this problem step by step, using logical reasoning and a programmatic approach to demonstrate the solution.

(A) Solution for when there's room for three boxes on the table... (python code)...

Let me break down the solutions:

(A) Solution for Three-Space Table:

1. Move Blue box to the table
2. Move Red box onto Blue box
3. Move Green box onto Red box
4. Move Blue box onto the table

Outcome:

-Blue box on the table

-Red box on the Green box

-Green box on the Blue box

(B) Solution for Two-Space Table:

This is more constrained and requires a different approach:

1. Move Blue box to the one open table space
2. Move Red box onto Green box (using the freed space)
3. Move Blue box onto Red box (now on Green)

The key insights are: . . .

The code demonstrates the logical steps to solve both scenarios, showing how to systematically move the boxes to achieve the desired configuration."

Evaluation: Running the code yielded "Solution for 3-space table:

Move Blue box to table

Move Red box off the table

Move Green box off the table

Move Blue box on table

Solution for 2-space table:

Move Blue box to table

Move Red box onto Green box

Move Blue box onto Red box"

The response was incorrect, since there was no solution for the second case given the constraints and that moves off the table were not part of the problem specification.

Problem 44. Claude Response: "I'll solve this Sudoku puzzle using a backtracking algorithm, which is an efficient method for solving Sudoku grids... (python code)...

Let me explain the solution approach:

1. Backtracking Algorithm:

-The solver uses a recursive backtracking method. . .

2. Validation Checks: . . .

3. Solution Process:

-Recursively tries to fill empty cells. . .

When you run this code, it will output the complete solved Sudoku grid where all empty cells (0s) are replaced with valid numbers that satisfy Sudoku rules."

Evaluation: Running the code yielded

"Solved Sudoku Grid:

3 1 6 5 7 8 4 9 2

5 2 9 1 3 4 7 6 8

4 8 7 6 2 9 5 3 1

2 6 3 4 1 5 9 8 7

9 7 4 8 6 3 1 2 5

8 5 1 7 9 2 6 4 3

1 3 8 9 4 7 2 5 6

6 9 2 3 5 1 8 7 4

7 4 5 2 8 6 3 1 9"

The solution was correct.

Problem 45. Claude Response: "I'll solve this logic puzzle by creating a systematic approach to determine the jobs of each person through deduction and constraint elimination... (python code)... Let me break down the solution approach:

1. Constraint Analysis:

-Each person holds exactly 2 jobs

- There are 8 unique jobs in total
- Several specific constraints are given

2. Solution Strategy:

- Generate all possible job assignments
- Check each assignment against the given constraints
- Find the unique solution that satisfies all conditions

3. Key Constraints Checked:...

4. Computational Approach:

- Uses combinatorics to generate possible job assignments...

When you run this, it will output the specific job assignment that resolves all the constraints in the puzzle."

Evaluation: The program output was

"Job Assignment Solution:

Roberta: chef, guard

Steve: chef, nurse

Thelma: chef, clerk

Pete: chef, police officer"

The solution was not correct according to the specification.

The overall correct answers using Claude were 9 out of 15.

3.3. Model Checking Using Gemini

The responses of Gemini to all problems were compared to the ground truth and the results are shown in the following (full sessions of all LLMs with comments are available at: <https://github.com/sbatsakis/LLM-modelchecking.git> (accessed on 1 December 2024)).

Problem 46. Gemini Response: "To determine the satisfiability of the given Boolean formula, we can use a systematic approach:

1. Analyze the constraints:...
2. Construct a truth table:

While a full truth table for 7 Boolean variables would be quite large, we can focus on the key implications from the above analysis.

```
x1 x2 x3 x4 x5 x6 x7 Formula Satisfied?
T F T F -T -Yes
T F F F -T -Yes
F T T F -T -Yes
F T F F -T -Yes
T T ----Yes (from the first two clauses)
--T --T -Yes (from the third clause)
---F --T Yes (from the fourth clause)
---F --F Yes (from the fifth clause)
```

3. Conclusion:

As we can see from the truth table, there are multiple assignments of truth values to the variables that satisfy the formula. Therefore, the formula is satisfiable.

Note: While this approach is effective for smaller formulas, for more complex Boolean formulas, automated SAT solvers are often used to determine satisfiability efficiently."

Evaluation: The response was correct regarding satisfiability. When asked for one such assignment that satisfies the formula, the response was "Here's one possible assignment that satisfies the given formula:

x1 = True

x2 = False

$x3 = \text{True}$
 $x4 = \text{False}$
 $x5 = \text{False (or True)}$
 $x6 = \text{True}$
 $x7 = \text{True (or False)}$

With this assignment, all clauses are satisfied:

1. $(x1 \vee x2 \vee x5 \vee x4) : \text{True}$
2. $(x1 \vee x2 \vee \neg x5 \vee x4) : \text{True}$
3. $(x3 \vee x6) : \text{True}$
4. $(\neg x4 \vee x7 \vee x1) : \text{True}$
5. $(\neg x4 \vee \neg x7 \vee x2) : \text{True}$

As you can see, there are multiple possible assignments that satisfy the formula due to the flexibility in the values of $x5$ and $x7$." The response was correct.

Problem 47. Gemini Response: "Let's analyze the statements: 1. A says: "B is a knave or C is a knave."

-If A is a knight, then this statement must be true. This means either B or C (or both) must be a knave.

-If A is a knave, then this statement must be false. This means both B and C must be knights.

2. B says: "A is a knight."

-If B is a knight, then A must be a knight.

-If B is a knave, then A must be a knave.

Combining these two statements and considering the possibilities, we can deduce the following:

Case 1: A is a knight.

-If A is a knight, then statement 1 is true. This means either B or C is a knave.

-If B is a knave, then statement 2 is false, which contradicts B being a knight.

-Therefore, B must be a knight.

-If B is a knight, then A's statement that "B or C is a knave" must be true.

-Since B is a knight, C must be a knave.

Conclusion:

-A is a knight.

-B is a knight.

-C is a knave.

Evaluation: The solution proposed by Gemini was correct.

Problem 48. Gemini Response: "Problem A: 3-Gallon and 5-Gallon Jugs

Solution:

1. Fill the 5-gallon jug completely.

2. Pour the water from the 5-gallon jug into the 3-gallon jug until the 3-gallon jug is full.

Now, the 5-gallon jug has 2 gallons of water left.

3. Empty the 3-gallon jug.

4. Pour the 2 gallons of water from the 5-gallon jug into the 3-gallon jug.

5. Fill the 5-gallon jug completely again.

6. Top off the 3-gallon jug with water from the 5-gallon jug. Since the 3-gallon jug already had 2 gallons, you will only need to add 1 gallon more.

7. Now, the 5-gallon jug will have exactly 4 gallons of water left.

Problem B: 3-Gallon and 9-Gallon Jugs

Solution:

1. Fill the 9-gallon jug completely.

2. Pour water from the 9-gallon jug into the 3-gallon jug until the 3-gallon jug is full. Now, the 9-gallon jug has 6 gallons of water left.

3. Empty the 3-gallon jug.
4. Pour 3 gallons of water from the 9-gallon jug into the 3-gallon jug.
5. Fill the 9-gallon jug completely again.
6. Top off the 3-gallon jug with water from the 9-gallon jug. Now, the 9-gallon jug has 7 gallons of water left.

Therefore, the 9-gallon jug now contains exactly 7 gallons of water."

Evaluation: Partly correct: The solution to A was correct, the response for B was not.

Problem 49. Gemini Response: "This is a classic cryptarithmic puzzle. Let's break down the problem: We have 8 unique letters: K, I, S, P, A, O, N. Each letter represents a unique digit from 0 to 9. We can start by looking at the ones place: $S * S = N$. This means N must be a perfect square. The only perfect squares between 0 and 9 are 0, 1, 4, and 9. However, N can't be 0 or 1 because that would make the product too small. So, N is either 4 or 9. Let's try $N = 4$:... However, solving this kind of puzzle manually can be quite time-consuming. A more efficient approach is to use a computer program to brute-force the solution by checking all possible combinations of digits for the letters. You can use a programming language like Python to implement this... While I can't provide a complete code implementation here due to its length and complexity, you can find many online resources and libraries that can help you with this task."

Evaluation: When asked to provide such code, Gemini responded with the following code. The program's response was "No solution found", which was incorrect.

Problem 50. Gemini Response: "I do not have enough information about that person to help with your request. I am a large language model, and I am able to communicate and generate human-like text in response to a wide range of prompts and questions, but my knowledge about this person is limited."

Evaluation: The LLM failed to address the problem.

Problem 51. Gemini Response: "To solve this problem, we can use a systematic approach. We need to find a combination of bags with coin counts from the given options that adds up to 100. Let's analyze the possible combinations:..."

Conclusion:

Based on the given coin counts, it's not possible to exactly determine the number of bags stolen to reach a total loss of 100 coins. There might be a constraint or missing information in the problem statement."

Evaluation: When asked explicitly to provide code, Gemini provided a Python script.

Running the code yielded the following:

"Possible combination of stolen bags: 2 bags of 16 coins 4 bags of 17 coins"

The solution was correct.

Problem 52. Gemini Response: "To solve this graph coloring problem, we can use a backtracking algorithm. Here's a Python implementation...(python code)..."

This code iteratively assigns colors to vertices, backtracking when a conflict arises. It continues until a valid coloring is found or all possibilities are exhausted. In this specific case, a valid coloring is possible. The algorithm will find a solution and print the assignment of numbers to vertices."

Evaluation: Running the code failed to provide a solution. So Gemini failed to solve this problem.

Problem 53. Gemini Response: "Yes, it is possible to place two queens and seven knights on a 6×6 chessboard without any pieces attacking each other.

Here's one possible configuration:

```
A B C D E F
1 Q . K . K .
```


2 . K . . K .
 3 K . . Q . K
 4 . K . K . .
 5 K . . K . .
 6 . . K . K .

In this configuration:

** The two queens are placed on squares A1 and C3.*

** The seven knights are placed on the remaining squares marked with 'K'.*

This arrangement ensures that no piece can attack another."

Evaluation: *The response was not correct regarding both the number of knights and the non-attacking restriction.*

Problem 54. Gemini Response: *“(A) Checking the Safety Property”*

To check if the property holds, we can analyze the state transitions and the semaphore mechanism.

Entering the Critical Section:

-A process can only enter the critical section if the semaphore is FALSE.

-Once a process enters the critical section, it sets the semaphore to TRUE, preventing the other process from entering.

Exiting the Critical Section:

-When a process exits the critical section, it sets the semaphore to FALSE, allowing the other process to enter.

*Based on this analysis, it is clear that **the safety property holds**. The semaphore mechanism ensures that only one process can be in the critical section at a time.*

“(B) Expressing the Safety Property in CTL”

The CTL formula for the safety property is:

$$AG\neg(\text{proc1.critical} \wedge \text{proc2.critical})$$

This formula states that it is always globally true that it is not the case that both processes are in the critical state simultaneously.

“(C) Checking the Liveness Property”

To check the liveness property, we need to ensure that a process that wants to enter the critical section will eventually do so.

In this case, the fairness constraint ensures that both processes will eventually get a chance to execute. Therefore, if a process is in the “entering” state, it will eventually acquire the semaphore and enter the critical section.

“(D) Expressing the Liveness Property in CTL”

The CTL formula for the liveness property is:

$$AG(\text{proc1.entering} \rightarrow AF\text{proc1.critical})$$

This formula states that it is always globally true that if process 1 is in the entering state, it will eventually enter the critical state.

A similar formula can be written for process 2.

Evaluation: *The translations to CTL were correct and so was the response for property one, Gemini did not provide a direct response for the second property, but using the translation the response could be found using specialized model checking tools such as NuXMV.*

Problem 55. Gemini Response: *“(A) Expressing the Safety Property in LTL”*

The LTL formula for the safety property that the two processes cannot be in the critical region at the same time is:

$$G\neg(\text{proc1.critical} \wedge \text{proc2.critical})$$

This formula states that globally, it is always the case that it is not the case that both processes are in the critical section simultaneously.

****Checking the Property:****

To formally verify this property, we would typically use a model checker or a theorem prover. These tools can analyze the state space of the system and determine if the property holds.

**** (B) Expressing the Liveness Property in LTL ****

The LTL formula for the liveness property that whenever a process wants to enter its critical section, it eventually does is:

$$G(\text{proc1.entering} \rightarrow F\text{proc1.critical})$$

This formula states that globally, it is always the case that if process 1 is in the entering state, it will eventually enter the critical state.

****Checking the Property:****

Again, we would use a model checker or a theorem prover to verify this property. The tool would analyze the state space to determine if the property holds for all possible execution paths.

By using these formal verification techniques, we can rigorously analyze the system and ensure that it satisfies the desired properties."

Evaluation: Both translations were correct.

Problem 56. Gemini Response: "The maximum number of bishops that can be placed on an 8x8 chessboard without attacking each other is ****14****."

Here's one possible configuration:

	A	B	C	D	E	F	G	H
1	B	.	B	.	B	.	B	.
2	.	B	.	B	.	B	.	B
3	B	.	B	.	B	.	B	.
4
5
6	B	.	B	.	B	.	B	.
7	.	B	.	B	.	B	.	B
8	B	.	B	.	B	.	B	.

As you can see, the bishops are placed diagonally, ensuring that no two bishops can attack each other."

Evaluation: The number was correct but the arrangement on the chessboard was not.

Problem 57. Gemini Response: "Here's how you can correctly label the boxes:

1. ****Identify the Incorrectly Labeled Box:****

* You know that no box is labeled correctly.

* So, if you open the box labeled "Pencils and Pens", and you take out a pencil, you know that this box ****cannot**** actually contain both pencils and pens.

* Therefore, this box must contain only pencils or only pens.

2. ****Correct the Labels:****

* Since the box labeled "Pencils and Pens" is incorrect, it must actually contain either only pencils or only pens.

* If you took out a pencil, the box must contain only pencils.
 * So, you can relabel this box as "Pencils."
 * Now, the box labeled "Pencils" must be incorrect, as it cannot contain only pencils.
 * Therefore, it must contain either only pens or both pencils and pens.
 * Since we know the box labeled "Pencils and Pens" only contains pencils, the box labeled "Pencils" must contain both pencils and pens.
 * By process of elimination, the remaining box, labeled "Pens", must contain only pens.
 By taking out just one object from the box labeled "Pencils and Pens", you can correctly identify the contents of all three boxes."

Evaluation: Although the first step was correct, there was then a problem in reasoning (i.e., the last step contradicts the assumption that all labels are wrong).

Problem 58. Gemini Response: "(A) Three-Box Table Yes, it's possible to rearrange the boxes as you described. Here's a step-by-step solution:

1. Move the Green Box:
-Lift the Green box off the table.
2. Move the Blue Box:
-Lift the Blue box off the Red box.
3. Place the Green Box on the Blue Box:
-Place the Green box onto the Blue box.
4. Place the Blue Box on the Table:
-Place the Blue box (with the Green box on top) onto the table.
5. Move the Red Box:
-Lift the Red box off the table.
6. Place the Red Box on the Green Box:
-Place the Red box onto the Green box.

Now, the arrangement is: Red on Green, Green on Blue, and Blue on the table.

(B) Two-Box Table No, it's not possible to achieve the desired arrangement with a two-box table.

The limitation of having only two spaces on the table restricts the possible moves. You can't place three boxes in a stacked configuration on a two-box table."

Evaluation: The response was correct regarding the second case, but in the first instance moves that were not implied by the problem statement were used.

Problem 59. Gemini Response: Here's the solved Sudoku puzzle:

```
{3, 1, 6, 5, 7, 8, 4, 9, 2},
{5, 2, 9, 1, 4, 3, 7, 6, 8},
{4, 8, 7, 6, 2, 9, 5, 3, 1},
{2, 6, 3, 7, 1, 5, 9, 8, 4},
{9, 7, 4, 8, 6, 3, 1, 2, 5},
{8, 5, 1, 4, 9, 2, 6, 7, 3},
{1, 3, 8, 9, 5, 7, 2, 5, 6},
{6, 9, 2, 3, 8, 1, 5, 7, 4},
{7, 4, 5, 2, 3, 6, 8, 1, 9}}
```

Evaluation: Gemini provided a solution for the Sudoku, but not all entries were correct.

Problem 60. Gemini Response: Let's analyze the clues: 1. Nurse is male: This narrows down the possibilities for the nurse to Steve or Pete.

2. Husband of the chef is the clerk: This means the chef is female.

3. Roberta is not a boxer: This eliminates boxing as a possibility for Roberta.

4. Pete has no education past the ninth grade: This eliminates jobs like teacher, clerk, and police officer for Pete. So, Pete must be the boxer or the actor.

5. Roberta, the chef, and the police officer went golfing: This means Roberta is not the chef or the police officer.

From the above clues, we can deduce the following:

-Roberta: Actor, Boxer

-Thelma: Chef, Police Officer

-Steve: Nurse, Clerk

-Pete: Boxer, Actor”

Evaluation: The solution was incorrect.

3.4. Summary of Results

The results of the evaluation are summarized in Table 1.

Table 1. Summary of results.

Problem	ChatGPT	Claude	Gemini
(1)	Correct	Correct	Correct
(2)	Incorrect	Correct	Correct
(3)	Partially correct	Correct	Partially correct
(4)	Incorrect	Incorrect	Incorrect
(5)	Incorrect	Incorrect	Incorrect
(6)	Correct	Correct	Correct
(7)	Correct	Correct	Incorrect
(8)	Incorrect	Incorrect	Incorrect
(9)	Correct	Correct	Correct
(10)	Correct	Correct	Correct
(11)	Incorrect	Incorrect	Incorrect
(12)	Correct	Correct	Incorrect
(13)	Partially correct	Incorrect	Partially correct
(14)	Correct	Correct	Incorrect
(15)	Incorrect	Incorrect	Incorrect

The results were correct if the solutions using the LLMs were identical to the correct solutions identified using the specialized mode checking tools and evaluated manually by the authors. Solutions were incorrect when they were different from the evaluated correct solutions (incorrectness was also verified manually by the authors in addition to the comparison of responses with the correct solutions from the model checking tools). Notice that, in case of some problems that consisted of two parts (Problems 3, 9, 10, and 13), the solutions of the LLMs could be correct for one part and incorrect for the other. In this case, the solution was characterized as partially correct.

The results indicate that LLMs are not yet reliable for dealing with model checking problems and that there were problems such as 4, 5, 8, 11, and 15 that all LLMs failed to solve. By manually inspecting the provided responses, many case were identified (including the responses of all LLMs for the abovementioned problems 4, 5, 8, 11, and 15) were one sentence did not logically follow from the previous ones or even contradicted previous ones. For example, in Problem 3 using ChatGPT, at step 8 of the second instance when the two jugs with capacities of 3 and 9 were full, the suggested action “Pour water

from the 9-gallon jug into the 3-gallon jug until the 3-gallon jug is full" cannot actually be applied, since the 3 gallon jug is already full, and according to ChatGPT this resulted in the nine gallon jug containing 7 seven gallons, which is clearly incorrect. Using Gemini for the same problem produced the same error at the same step as ChatGPT. In the case of Problem 11 using Claude, notice in the output that in step 3 the response contained the statement "When you run this code, it will output: -Maximum number of bishops (which is 14 for an 8×8 board)..." and when running the code the output contained the positions of 8 bishops with attacks between them, contradicting the previous statement in the output. Thus, all examined LLMs contained basic reasoning errors in their responses, indicating that they lack robust logical capabilities.

Overall, Claude achieved the highest performance, with 9 out of 15 problems solved, followed closely by ChatGPT with 7 correct and 2 partially correct solutions, followed by Gemini with 5 correct and 2 partially correct solutions. Providing python scripts for solving problems (which was the typical response by Claude) was on average more accurate than directly solving the problems, but also more time consuming for the user (considering the additional time for running the generating script to obtain the solution). The most common response type by Gemini was solving the problems directly, while the responses of ChatGPT contained a combination of source code and direct solutions.

The exact reason for the reported performance is difficult to analyze, due to the black-box architecture of large LLMs having parameters in the order of trillions, which is often reported as a fundamental problem of LLMs. In addition, the overall approach of building text generating systems upon vast textual training sets seems to have limitations when faced with problems requiring accurate reasoning capabilities, especially in critical applications such as medicine. Whether this is a limitation that can be dealt with by improving the capabilities of LLMs, while retaining their generic design and architecture, or whether this requires a major design shift is an open question.

4. Conclusions and Topics for Future Research

This is a first step towards understanding the reasoning capabilities of large language models for model checking. The results using ChatGPT, Claude, and Gemini showed a mixed picture: several problems were solved correctly, mainly by providing a sound Python program or an accurate translation to temporal logic formulas, but on the other hand, the tested LLMs failed to solve other problems. It is noteworthy to point out the fact that, in some example problems, all LLMs failed, illustrating the reasoning limitations of current LLMs.

Since LLMs are one of the most active areas of research, future work will include updating the current work with the addition of new evaluation problems, aiming to create larger benchmarks for LLM model checking evaluation. Furthermore, future work will include evaluating new LLMs or updated versions of existing LLMs, which appear at a fast pace, so as to keep track of their capabilities for model checking.

Another direction of future work, in addition to the evaluation of LLMs for model checking, will be the improvement of their performance on this task. Given that efficient model checking systems such as Pro B already exist and were used in this work for providing the golden standard for evaluation, a future direction could be the deployment of hybrid systems that combine and integrate the LLMs with specialized model checking tools. In such systems, the LLM could recognize the type of problem and generate an input compatible with the format required by the specialized model checking system; then, the model checking tool could be used to generate the solution; finally, the LLM could be employed again to convert the solution of the model checking tool to a natural language format that is understandable by the end-users. Such a hybrid system would combine the

accuracy of specialized model checking systems with the capabilities of LLMs in handling and generating natural language, allowing usage by users that are not familiar with the complexities of specialized reasoning systems.

Overall, there is a need for a systematic analysis of reasoning schemes, chains of reasoning, etc. Developing relevant benchmarks is an important vehicle to this end and we intend to work on this. Our initial work, reported here, was about model checking but we have longer-term plans to investigate and benchmark other forms of reasoning, including first-order reasoning, epistemic reasoning, reasoning about change, reasoning about action, and reasoning about time.

Author Contributions: Conceptualization, S.B., I.T., M.M., N.P. and G.A.; methodology, S.B., I.T., M.M., N.P. and G.A.; software, S.B., I.T. and M.M.; validation, S.B., I.T., M.M., N.P. and G.A.; formal analysis, S.B., I.T., M.M., N.P. and G.A.; investigation, S.B., I.T., M.M., N.P. and G.A.; resources, S.B., I.T. and M.M.; data curation, S.B. and I.T.; writing—original draft preparation, S.B., I.T. and M.M.; writing—review and editing, S.B., I.T., M.M., N.P. and G.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding

Data Availability Statement: Data available at: <https://github.com/sbatsakis/LLM-modelchecking.git> (accessed on 1 December 2024).

Conflicts of Interest: The authors declare no conflict of interest.

References

- Thoppilan, R.; De Freitas, D.; Hall, J.; Shazeer, N.; Kulshreshtha, A.; Cheng, H.T.; Jin, A.; Bos, T.; Baker, L.; Du, Y.; et al. Lamda: Language models for dialog applications. *arXiv* **2022**, arXiv:2201.08239.
- OpenAI. GPT-4 Technical Report, 2023. *arXiv* **2023**, arXiv:2303.08774. [CrossRef]
- Lambert, N.; Castriato, L.; von Werra, L.; Havrilla, A. Illustrating Reinforcement Learning from Human Feedback (RLHF). *Hugging Face Blog*. 2022. Available online: <https://huggingface.co/blog/rlhf> (accessed on 1 December 2024).
- Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.L.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. Training language models to follow instructions with human feedback, 2022. *arXiv* **2022**, arXiv:2203.02155. [CrossRef]
- Letters, F.O. Pause giant AI experiments: An open letter. *Future of Life Institution*. **2023**. Available online: <https://futureoflife.org/open-letter/pause-giant-ai-experiments> (accessed on 1 December 2024).
- Bubeck, S.; Chandrasekaran, V.; Eldan, R.; Gehrke, J.; Horvitz, E.; Kamar, E.; Lee, P.; Lee, Y.T.; Li, Y.; Lundberg, S.; et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv* **2023**, arXiv:2303.12712.
- Luccioni, A.S.; Viguier, S.; Ligozat, A.L. Estimating the Carbon Footprint of BLOOM, a 176B Parameter Language Model. *arXiv* **2022**, arXiv:2211.02001.
- Strubell, E.; Ganesh, A.; McCallum, A. Energy and Policy Considerations for Deep Learning in NLP. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; pp. 3645–3650.
- Luccioni, A.; Viviano, J. What’s in the box? an analysis of undesirable content in the Common Crawl corpus. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), Online, 1–6 August 2021; pp. 182–189.
- Bowman, S.R. Eight things to know about large language models. *arXiv* **2023**, arXiv:2304.00612.
- Bills, S.; Cammarata, N.; Mossing, D.; Tillman, H.; Gao, L.; Goh, G.; Sutskever, I.; Leike, J.; Wu, J.; Saunders, W. Language Models Can Explain Neurons in Language Models. 2023. Available online: <https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html> (accessed on 1 December 2024).
- Bender, E.M.; Gebru, T.; McMillan-Major, A.; Shmitchell, S. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? In Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, Virtual, 3–10 March 2021; pp. 610–623.
- Zhang, H.; Li, L.H.; Meng, T.; Chang, K.W.; Broeck, G.V.d. On the paradox of learning to reason from data. *arXiv* **2022**, arXiv:2205.11502.

14. Antoniou, G.; Batsakis, S. Defeasible Reasoning with Large Language Models—Initial Experiments and Future Directions. In Proceedings of the CEUR Workshop Proceedings. CEUR Workshop Proceedings, Kyiv, Ukraine, 30 November 2023; Volume 3485, p. 7687.
15. Cao, L. Enhancing reasoning capabilities of large language models: A graph-based verification approach. *arXiv* **2023**, arXiv:2308.09267.
16. Huang, J.; Chang, K.C.C. Towards Reasoning in Large Language Models: A Survey. *arXiv* **2022**, arXiv:2212.10403.
17. Mirzadeh, I.; Alizadeh, K.; Shahrokhi, H.; Tuzel, O.; Bengio, S.; Farajtabar, M. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv* **2024**, arXiv:2410.05229.
18. Glazer, E.; Erdil, E.; Besiroglu, T.; Chicharro, D.; Chen, E.; Gunning, A.; Olsson, C.F.; Denain, J.S.; Ho, A.; Santos, E.d.O.; et al. Frontiermath: A benchmark for evaluating advanced mathematical reasoning in ai. *arXiv* **2024**, arXiv:2411.04872.
19. Guo, Z.; Jin, R.; Liu, C.; Huang, Y.; Shi, D.; Yu, L.; Liu, Y.; Li, J.; Xiong, B.; Xiong, D.; et al. Evaluating large language models: A comprehensive survey. *arXiv* **2023**, arXiv:2310.19736.
20. Chang, Y.; Wang, X.; Wang, J.; Wu, Y.; Yang, L.; Zhu, K.; Chen, H.; Yi, X.; Wang, C.; Wang, Y.; et al. A survey on evaluation of large language models. *ACM Trans. Intell. Syst. Technol.* **2024**, *15*, 39. [[CrossRef](#)]
21. Zhang, Y.; Mao, S.; Ge, T.; Wang, X.; de Wynter, A.; Xia, Y.; Wu, W.; Song, T.; Lan, M.; Wei, F. LLM as a Mastermind: A Survey of Strategic Reasoning with Large Language Models. *arXiv* **2024**, arXiv:2404.01230.
22. Siddiq, M.L.; Da Silva Santos, J.C.; Tanvir, R.H.; Ulfat, N.; Al Rifat, F.; Carvalho Lopes, V. Using large language models to generate junit tests: An empirical study. In Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering, Salerno, Italy, 18–21 June 2024; pp. 313–322.
23. Wang, X.; Wei, J.; Schuurmans, D.; Le, Q.; Chi, E.; Narang, S.; Chowdhery, A.; Zhou, D. Self-consistency improves chain of thought reasoning in language models. *arXiv* **2022**, arXiv:2203.11171.
24. Xie, S.; Liu, R.; Wang, X.; Luo, X.; Sugumaran, V.; Yu, H. Hierarchical Knowledge-Enhancement Framework for multi-hop knowledge graph reasoning. *Neurocomputing* **2024**, *588*, 127673. [[CrossRef](#)]
25. Wang, Y.; Xia, N.; Yu, H.; Luo, X. Knowledge Graph Reasoning via Dynamic Subgraph Attention with Low Resource Computation. *Neurocomputing* **2024**, *595*, 127866. [[CrossRef](#)]
26. Zhen, C.; Shang, Y.; Liu, X.; Li, Y.; Chen, Y.; Zhang, D. A Survey on Knowledge-Enhanced Pre-trained Language Models. *arXiv* **2022**, arXiv:2212.13428.
27. Yin, D.; Dong, L.; Cheng, H.; Liu, X.; Chang, K.W.; Wei, F.; Gao, J. A survey of knowledge-intensive nlp with pre-trained language models. *arXiv* **2022**, arXiv:2202.08772.
28. Kaplan, J.; McCandlish, S.; Henighan, T.; Brown, T.B.; Chess, B.; Child, R.; Gray, S.; Radford, A.; Wu, J.; Amodei, D. Scaling Laws for Neural Language Models, 2020. *arXiv* **2020**, arXiv:2001.08361. [[CrossRef](#)]
29. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models are Few-Shot Learners, 2020. *arXiv* **2020**, arXiv:2005.14165. [[CrossRef](#)]
30. Clarke, E.M. Model checking. In Proceedings of the Foundations of Software Technology and Theoretical Computer Science: 17th Conference, Kharagpur, India, 18–20 December 1997; Proceedings 17; Springer: Berlin/Heidelberg, Germany, 1997; pp. 54–56.
31. Rozier, K.Y. Linear temporal logic symbolic model checking. *Comput. Sci. Rev.* **2011**, *5*, 163–203. [[CrossRef](#)]
32. Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F.L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. Gpt-4 technical report. *arXiv* **2023**, arXiv:2303.08774.
33. Team, G.; Anil, R.; Borgeaud, S.; Alayrac, J.B.; Yu, J.; Soricut, R.; Schalkwyk, J.; Dai, A.M.; Hauth, A.; Millican, K.; et al. Gemini: A family of highly capable multimodal models. *arXiv* **2023**, arXiv:2312.11805.
34. Leuschel, M.; Butler, M. ProB: A model checker for B. In Proceedings of the FME 2003: Formal Methods: International Symposium of Formal Methods Europe, Pisa, Italy, 8–14 September 2003; Proceedings; Springer: Berlin/Heidelberg, Germany, 2003; pp. 855–874.
35. Smullyan, R. *What is the Name of this Book?* Touchstone Books: Guildford, UK, 1986.
36. Boldi, P.; Santini, M.; Vigna, S. Measuring with jugs. *Theor. Comput. Sci.* **2002**, *282*, 259–270. [[CrossRef](#)]
37. Pelletier, F.J. Seventy-five problems for testing automatic theorem provers. *J. Autom. Reason.* **1986**, *2*, 191–216. [[CrossRef](#)]
38. Murty, K.G. *Optimization Models for Decision Making*; University of Michigan: Ann Arbor, MI, USA, 2003.
39. Cavada, R.; Cimatti, A.; Keighren, G.; Olivetti, E.; Pistore, M.; Roveri, M. NuSMV 2.2 Tutorial. *ITC-Irst Sommarive* **2004**, *18*, 38055.
40. Gupta, N.; Nau, D.S. On the complexity of blocks-world planning. *Artif. Intell.* **1992**, *56*, 223–254. [[CrossRef](#)]
41. Wos, L.; Overbeek, R.; Lusk, E.; Boyle, J. *Automated Reasoning Introduction and Applications*; McGraw-Hill: Inc.: New York, NY, USA, 1992.
42. McCune, W. *Otter 3.3 Reference Manual*; Argonne National Laboratory: Argonne, IL, USA, 2003.
43. Shapiro, S.C. The jobs puzzle: A challenge for logical expressibility and automated reasoning. In Proceedings of the 2011 AAAI Spring Symposium, Technical Report SS-11-06, Stanford, CA, USA, 21–23 March 2011.

44. Cimatti, A.; Clarke, E.; Giunchiglia, E.; Giunchiglia, F.; Pistore, M.; Roveri, M.; Sebastiani, R.; Tacchella, A. Nusmv 2: An opensource tool for symbolic model checking. In Proceedings of the Computer Aided Verification: 14th International Conference, CAV 2002, Copenhagen, Denmark, 27–31 July 2002; Proceedings 14; Springer: Berlin/Heidelberg, Germany, 2002; pp. 359–364.
45. Cavada, R.; Cimatti, A.; Dorigatti, M.; Griggio, A.; Mariotti, A.; Micheli, A.; Mover, S.; Roveri, M.; Tonetta, S. The nuXmv symbolic model checker. In Proceedings of the Computer Aided Verification: 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, 18–22 July 2014; Proceedings 26; Springer: Berlin/Heidelberg, Germany, 2014; pp. 334–342.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.