

Leveraging Large Language Models as an Interface to Conflict Resolution for Human-AI Alignment in Air Traffic Control

Charles Berro*, Fotini Deligiannaki*, Thomas Stefani†, Johann Maximilian Christensen*,
Ingrid Gerdes‡, Frank Köster*, Sven Hallerbach† and Arne Raulf*

* *Institute for AI Safety and Security*
German Aerospace Center (DLR)
Sankt Augustin, Germany

† *Institute for AI Safety and Security*
German Aerospace Center (DLR)
Ulm, Germany

‡ *Institute for Flight Guidance*
German Aerospace Center (DLR)
Braunschweig, Germany

Email: charles.berro@dlr.de, fotini.deligiannaki@dlr.de, thomas.stefani@dlr.de, johann.christensen@dlr.de,
ingrid.gerdes@dlr.de, frank.koester@dlr.de, sven.hallerbach@dlr.de and arne.raulf@dlr.de

Abstract—A steep aircraft increase is forecasted in the near future, putting additional strain on en-route air traffic control. To meet the safety and efficiency goals, contemporary research explores the Single Controller Operations (SCOs) concept to replace traditional positioning of two Air Traffic Controllers (ATCOs) per sector. During workshops with ATCOs addressing SCOs, Conflict Resolution (CR) has been identified as one important task that can be supported by automation. Although existing work on CR solvers shows promising results, solvers based on fixed optimization functions are incompatible with the dynamic evolving preferences of ATCOs. This work proposes two additional steps that filter and rank CR solutions based on a set of rules in natural language—forming a flexible policy—to better align with ATCO preferences in automation-supported CR. Inspired from related work on LLM-driven agents, an algorithm using LLMs to filter and sort CR solutions for alignment with natural language policies is presented. The algorithm is tested on a synthetic dataset of policies and solutions for several minimal filtering and sorting scenarios. The experiments show success in solving the task in most cases and a correct understanding of the task by the LLM. Nevertheless, the analysis of failure cases highlights several limitations of LLMs that must be considered in future research and development of similar systems.

Index Terms—Air Traffic Control, Single Controller Operation, Large Language Models

I. INTRODUCTION

The increasing complexity in air traffic control (ATC) as a consequence of rising traffic demands and a persistent shortage of qualified Air Traffic Controllers (ATCOs) present mounting challenges for maintaining safety and efficiency in en-route ATC [1]. To address these issues, automation has been a longstanding component of ATC working environments, initially supporting human controllers through information processing and visualization tools [2]. However, as these tools reach their functional limits, a paradigm shift towards Artificial Intelligence (AI) based decision support is emerging as a promising solution [3].

At the core of this shift lies the challenge of effective interaction between humans and AI-based systems [4]. In safety-critical and time-sensitive environments such as ATC, AI must not only perform its tasks competently but also

interact with human operators in a manner that preserves trust, transparency, and operational control [5]. This requires AI-based systems to be explainable, predictable, and capable of aligning their actions with the mental models and goals of the human ATCOs.

Large Language Models (LLMs) have recently shown great potential in bridging the communication and closing the gap between humans and AI-based systems [6]. Their capacity for natural language understanding and text generation positions them as a potential interface between human ATCOs and complex automated decision-support systems, especially in dynamically changing environments. By enabling intuitive, contextual, and reciprocal communication, LLMs can facilitate mutual understanding and enhanced situational awareness, key components of successful interaction between AI-based systems and humans.

This paper explores how LLMs can be leveraged as an interface between ATCOs and Conflict Resolution (CR) tools within the en-route ATC domain. Through filtering and sorting tasks based on dynamic policies from ATCOs, the algorithm overcomes the limitations of classical CR solvers bounded to fixed optimization functions and therefore allows for CR automation with dynamic rules (cf. Figure 1).

Section II gives an overview of existing CR solvers for en-route ATC and their role in the Single Controller Operations (SCOs) concept, as well as related works on the use of LLMs in the context of ATC. While some research efforts on a Digital Interactive Reliable Controller (DIRC) [7], [8] show how CR can be solved with predefined optimization functions, dealing with dynamic rules in natural language is not addressed. In Section III, the limitations of classical CR solvers are highlighted, and an algorithm using LLMs is proposed to overcome these limitations. As LLMs have major drawbacks, measures to support safety-by-design are presented in Subsection III-A. A prototype of the proposed algorithm is implemented and tested on a synthetic dataset. The results of these experiments are presented in Section IV.

The scope of the provided solution lays in fundamental

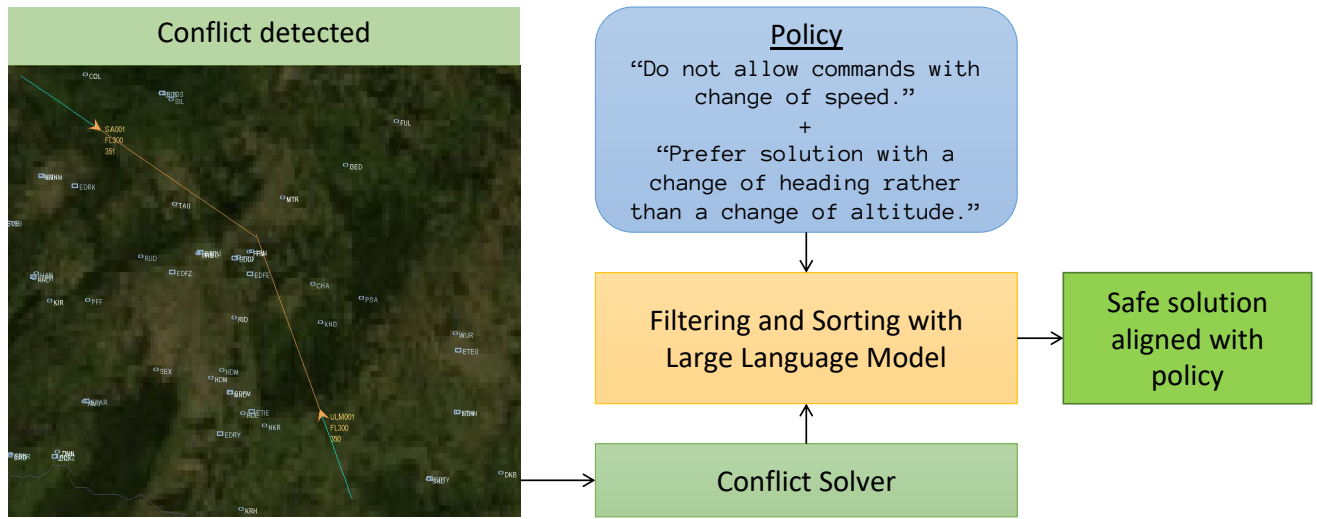


Figure 1: Basic concept of large language models as the interface to conflict solver to align with natural language policy.

research at low technology readiness levels to gather first experiences with AI scientists, ATC researchers and ATCOs.

II. RELATED WORK

A. Conflict Resolution for en-route Air Traffic Control

An important part of the work of ATCOs is the detection and resolution of conflicts between flights within their assigned sector [9]. During Conflict Detection (CD) the traffic is constantly monitored to identify upcoming conflicts caused by wrong planning, deviations from ATCO instructions or deviations from planned flight trajectories in position, speed, and flight level. If a conflict is detected, it is up to the responsible ATCO to perform Conflict Resolution (CR) by assigning a new conflict-free trajectory as a solution to this situation. To perform CR, ATCOs can be supported by automation. Examples of CR support include solution probes that present a list of feasible conflict-free trajectory options, or graphical interfaces on the radar screen that allow controllers to manipulate trajectories directly with real-time warnings if the proposed changes would introduce new conflicts. Developing methods for an automatic and reliable calculation of conflict-free solutions is therefore not only a highly safety-critical task but also an important research topic, especially for long and medium-term forecasts. An overview of conflict resolution methods can be found in other works [10]. These methods are often combined with probabilistic approaches [11] and 4-D trajectories [12]. Recent research tends to focus on applying mixed-integer linear program methods for solving conflicts [13], [14] or use machine learning methods such as reinforced learning [15].

As a variety of solutions might exist for a single conflict, the final choice happens based on requirements such as airspace safety, aircraft physical constraints, ATC rules, drawbacks of the solution, or ATCO preferences. Some requirements can be static while others can dynamically influence the choice of a solution. Conflict resolution solvers based on static optimization functions are not able to adapt to these dynamically evolving requirements. This requires a conflict resolution tool that is capable of adapting a solution based on requirements such as shorter trajectories, reaching the sector boundary on time, capable climb and descent rates, or sticking

close to the original trajectory. For the DLR projects DIAL¹ and LOKI² such a tool named TraGAT (Trajectory Generation and Advisory Tool) was implemented.

For the CR part of TraGAT, an evolutionary algorithm with a configurable evaluation function composed of selectable sub-functions for each evaluation aspect is implemented. These sub-functions are created in such a way that the evaluation function is always minimized. As degrees of freedom, the evolutionary algorithm may change speed, heading, flight level, or number and position of waypoints.

In conclusion, there are several CR solvers for en-route ATC which can provide conflict-free solutions, with TraGAT being an example of a configurable solver that returns multiple solutions to one conflict. Even the configurable CR solvers are bounded to predefined optimization (sub-)functions.

B. Human-AI Teaming in Air Traffic Control

Air Traffic Control sectors across Europe are typically staffed with two air traffic controllers. This team effort ensures an even distribution of task load, maintains safety through the four-eyes principle, and guarantees that each sector operates within its declared capacity, such as aircraft per hour [16]. While there is no regulatory requirement mandating two ATCOs per sector, a single ATCO is only deployed when traffic levels are forecasted to remain below half of the declared capacity. Before implementing SCOs, sectors are typically merged to balance staffing needs. However, ATCO scheduling cannot be adjusted for short-term operations, leading to occasional over-staffing due to fluctuating and unpredictable traffic patterns. As a result, single ATCO deployment occurs mainly during low-traffic periods such as nighttime, or in response to temporary controller shortages. Despite these operational strategies, ATCO shortages remain a persistent issue, contributing to air traffic delays [1], [17], [18]. Recruiting additional controllers is challenging given the demanding nature of the job and stringent selection criteria [19]. Meanwhile, air traffic is expected to increase further leading to continuous staffing difficulties. The concept of SCOs through human-AI teaming therefore gains relevance.

¹Der individuelle und automatisierte Luftverkehr

²Kollaboration von Luftfahrt-Operateuren und KI-Systemen

For SCOs to be viable without significantly reducing declared capacity or increasing ATCO workload further automation is essential [20]. Current ATC support tools such as Short-Term Conflict Alert and Medium-Term Conflict Detection primarily focus on information processing and visualization, requiring active engagement from controllers [21]–[23]. However, these tools do not fundamentally reduce workload or decision-making demands. AI-based solutions offer promising potential to reduce workload for ATCOs due to their capacity for solving and taking over even complex tasks [24], [25]. A major challenge in implementing AI-based ATC automation lies on the one hand in the lack of safety of said AI models, and on the other hand, in determining how to delegate tasks between human controllers and AI-based systems, and ensuring effective interaction between them [26]. The allocation of responsibilities must consider the strengths and weaknesses of both human operators and AI, creating a balance that maintains safety, efficiency, and trust in the system.

In workshops with ATCOs [27], conflict detection and resolution have been identified as crucial tasks when talking about AI-based support from a digital ATCO for reducing workload. For Human-AI teaming the ATCOs proposed an implementation of the conflict resolution task where several solutions are created and evaluated by the digital ATCO. Only the *best* should be presented to the human ATCO which they then can accept or reject, whereby the best is identified following ATCO-specific preferences. However, *best* in this context is rather subjective depending on ATCO preferences and given circumstances influenced by numerous factors, such as weather or traffic situation. Therefore, this task cannot be solved by classical optimization-based scores which are bounded to predefined optimization functions.

Large Language Models present a promising solution to this challenge as they can facilitate natural and intuitive communication between ATCOs and automation. By enabling contextual understanding, adaptive decision support, and real-time explanations, LLMs can serve as an interface for seamless human-AI teaming, ensuring that controllers remain in control while benefiting from AI-based assistance [28], [29].

Current research focuses on the concept of a digital interactive reliable controller (DIRC) which aims to enable AI-based automation in ATC [7], [8], [30]. DIRC centralizes interactions between the ATCO and system components, alleviating the need for ATCOs to manually cross-check, oversee, and activate multiple systems. This approach aims to reduce cognitive load and preventing ATCOs from being overwhelmed by system management tasks. Given the distinct strengths and limitations of AI and human controllers, implementing SCOs effectively will require a revised task allocation model that differs from the current executive-planner controller structure. A key aspect in fostering acceptance of AI-driven automation is the combination of natural language communication with interactive and reciprocal information exchange, ensuring that ATCOs feel engaged, informed, and confident in their collaboration with AI systems [31].

C. Application of LLMs in Air Traffic Control

In recent years LLMs have been applied in ATC in a series of research activities. Their benefits have been shown primarily in the area of information extraction from historical documents such as ground delay programs, for enhancing the

communication safety between pilots and ATCOs, and for reducing errors in speech commands due to misinterpreted verbal information [4], [29], [32]–[34]. Conversational LLM-based tools in other works [35] have also been proven successful by assisting decision-making during strategic air traffic management and re-planning while being restricted in their application only in non-safety critical tasks due to their limitations.

LLMs have recently been employed in conflict detection where they solve reasoning and tool selection tasks [36]. Specifically, the LLM agent monitors the air traffic as well as solves existing conflicts. Although the authors use a vector database of labeled solved and unsolved conflicts to guide and enhance the safety of LLM-based conflict resolution, this approach still presents substantial safety risks due to hallucinations. Precisely this issue, analyzed in upcoming sections, is being addressed in this paper by using LLMs to evaluate rather than generate or modify conflict resolutions. The aforementioned work as well as this paper initialize the investigation of using LLMs in reasoning tasks within ATC.

III. CONFLICT RESOLUTION UNDER POLICY

As shown in the related work section there exist numerous CR solvers for en-route ATC. In this section, two additional steps to filter and sort the results given by a CR solver are described. In these steps an LLM is used to verify the alignment between a solution and the policies defined by ATCO in natural language are described. This process aims to resolve the lack of flexibility of classical CR solvers. The proposed method is CR-solver agnostic as it does not depend on inner mechanisms of the CR solver.

A solution s produced by a CR solver is defined as a list of *commands* $s = [c_0, \dots, c_n]$, with c_i representing the i th command. The commands are restricted to a change of heading, speed, or altitude, where every command has its associated execution time. Although a proven-safe conflict resolution system is expected to only produce safe solutions s , a safe solution might not be a *good* solution. A good solution can have measurable elements such as the minimal distance to others, the number of commands it involves, and the added route length. These variables can be optimized to reduce the potential future risk and maximize the efficiency of conflict resolution. Specifically, the minimal distance to other aircraft has to comply with the regulatory separation standards. In addition, the number of commands should be minimized to reduce the resolution's complexity. Similarly, the added length of the new route should be penalized to reduce arrival delays and the flight's carbon footprint. Classical optimization and predefined rules can determine how good a solution is based on metrics and statically defined weights for the above elements. Nevertheless, classical optimization problems must be written in mathematical expressions or a programming language. Writing these expressions requires meticulous reflection time, mathematical expertise, and significant testing time, which is unsuitable for dynamically changing policies formulated by ATCOs. In the ideal case, good solutions should be identified based on dynamically evolving policies specified in natural language by the experienced ATCO as explained in Section II.

Policies may be dynamic with variability in their temporal scope. Examples include:

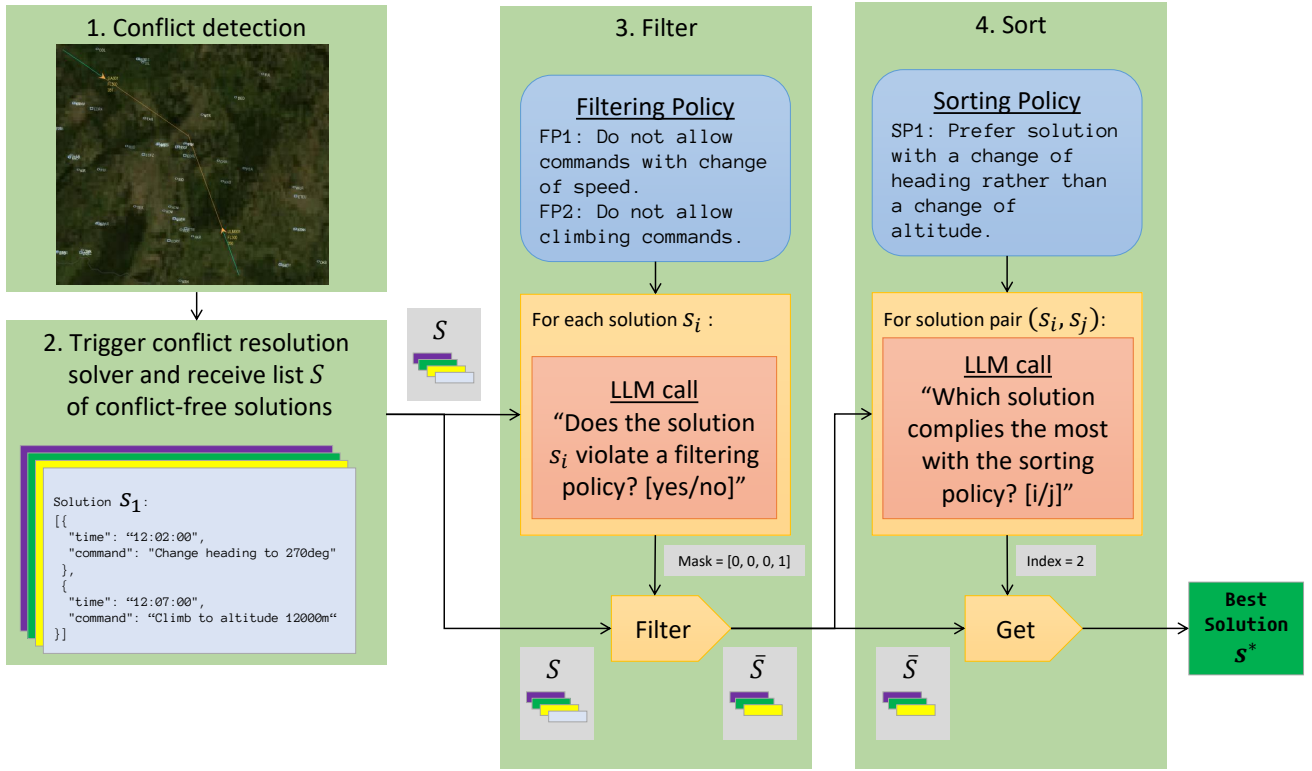


Figure 2: Filtering and sorting of conflict resolution solutions based on natural language policies. Once a conflict is detected a list S of diverse conflict-free solutions is produced by a trusted conflict solver. For each solution, an LLM assesses if the solution violates a filtering policy. The parsed LLM answers form a filtering mask, used to filter the list S into $\bar{S}$. For solution pairs in $\bar{S}$, the LLM estimates which solution is most complying with the sorting policy. Based on these comparisons, the index of the best solution in $\bar{S}$ is identified and the best solution s^* is returned. LLM calls are only performed in the mentioned steps (orange boxes).

- **Long-term policies:** "Handover from sector A to sector B should, whenever possible, be conducted via waypoint ABC at Flight Level 123 or above." (P1)
- **Short-term policies:** "Due to exceptional meteorological conditions, until 12:00 today, avoid delaying westbound flights to ensure they land before weather deterioration." (P2)
- **Very short-term:** "Aircraft ABC123 in sector A is unresponsive. Do not allow additional aircraft to enter sector A." (P3)

In the context of hybrid ATC the above policies are examples of tasks that could be partially covered by the digital ATCO, while the human ATCO focuses on critical resolutions. Examples of critical resolutions that should be handled by a human ATCO, e.g., heavy cargo flights that cannot comply with the flight level defined in (P1) because of very slow climbing rate; rerouting a delayed flight to avoid dangerous weather notified in (P2); or clearing the airspace around unresponsive flights notified in (P3).

Policies can be specified in different forms such as pseudo-code, mathematical expressions, or coding blocks. Using natural language is the most intuitive way as it is the current communication form used by ATCOs to exchange preferences with their teammates. The clear disadvantage of policies specified in natural language is that there is no straightforward translation to machine-interpretable concepts. Natural language processing, the branch of research exploring machine interpretation of natural language, is vast and its evolution is accelerating rapidly. Within this field, LLMs seem

like a suitable candidate to interpret ATC policies written in natural language.

The following section presents one approach to post-process CR solvers' outputs to comply with natural language policies.

A. Approach

The idea of an LLM-based text generation under constraints—defined as natural language policies—can be induced with a method similar to constitutional AI [37]. A constitution, consisting of a list of rules referred to in this paper as *policy*, is given to the LLM to assess whether or not a text entry is compliant. The policy can be used to filter not just one entry but a list of entries [38]. The proposed system leverages this idea with the LLM filtering suitable solutions based on policies. In this work, the LLM *filtering* idea is extended to an LLM *sorting* concept which consists of comparing different solutions based on a policy (cf. Subsection III-B). Since hallucinations are inevitable once LLMs are used as general problem solvers [39], [40], it is very important to note that solutions are neither created nor modified by the LLM. The outputs of the LLM are parsed into either a Boolean during filtering, or an integer during sorting, pointing at the index of the best solution when sorting several solutions (cf. Subsection III-B). This is a crucial design decision contributing to a system with safety-by-design.

In order to guide the LLM towards using the policy several prompt engineering methods are used. A method called Chain-of-Thought [41] which enforces a reasoning step has been proven to considerably improve reasoning

capabilities in LLMs. Notably, the reasoning has to be oriented as an *explanation* that can be used for explainability of the generated outputs. Another key prompting technique, inspired by ReAct [42], is to provide observations as *facts* before the reasoning step. The generation is guided to follow a specific answer format, as a *one-shot* example. The answer format is strictly defined and the chain-of-thought thoroughly guided, a widely used safeguarding mechanism for reducing hallucinations [39]. Reducing hallucinations is important to avoid wrongful reasoning by the LLM. The consequence of wrongful reasoning or other LLM failure would be picking a solution that does not align with ATCO policies, although still being safe since every solution in the solutions set is safe. This constitutes the second crucial design decision contributing to a system with safety-by-design.

The design decisions taken to improve the safety of the system are far from safety guarantees but rather contributions to future developments towards a safe system. In addition, the safety approaches bring technical safety, but do not cover the uncertainty which would be introduced by a less-than-best solution. The impact of this uncertainty requires a more human-factor-oriented approach, which is outside the scope of this work.

B. Algorithm

The filtering and sorting of CR solutions based on natural language policies is accomplished by the following steps which are explained in detail within the next paragraphs. A conflict is first detected which then triggers a conflict solver to generate a list S of safe solutions. The solutions list is then filtered and finally, pairs in the list $\bar{S}$ of filtered solutions are compared to identify the index of the overall best solution s^* (cf. Figure 2). During the filtering and sorting process the solutions are never modified.

A detected conflict between two aircraft contains several details such as the call sign of two aircraft, the Closest Point of Approach (CPA), the time and distance to CPA and horizontal relative heading angle. Based on the CD information, the CR solver provides a list S of safe solutions as described in Subsection III-A.

Filtering is accomplished by going through each solution one by one. For each solution s_i in S , a prompt is generated (see example in Section A). The goal of the prompt is to bias the LLM into predicting whether or not s_i is violating any of the filtering policies FP_i . The parsed answer must be a code block with a valid JSON containing one key “Answer” which value must be one of the two acceptable values “yes” and “no”. If the answer value is “yes”, one of the filtering policies has been violated and the solution s_i is discarded and not included in the filtered solutions list $\bar{S}$. Otherwise, if the answer value is “no”, none of the filtering policies has been violated and the solution s_i is included in the filtered solutions list $\bar{S}$. If the answer does not fit the required format or value, the answer is considered invalid. In this situation, a simple strategy is to discard the solution. More complex strategies to re-prompt the LLM can be elaborated but were not covered in this work. If the filtering policies are too restrictive, resulting in filtering all solutions, it is possible to search for a solution which partially aligns with the policies. Assuming that the policies are ordered by importance, the filtering process can be restarted with the last policy removed. It would be mandatory

to warn the ATCO that only partial alignment was achieved by referencing the policy that was not fulfilled.

Sorting starts with the initialization of the best solution s^* . The best solution can be arbitrarily chosen, randomly chosen, or chosen based on the scoring of the solution provided by the CR solver. The initialization choice does not impact safety (cf. Subsection III-A). For each solution s_i in $\bar{S}$, a prompt is generated (see example in Section A). The goal of the prompt is to bias the LLM into predicting which solution in (s^*, s_i) is the most complying with all sorting policies SP_i . The parsed answer must be a code block with a valid JSON containing one key “Answer” which value must be one of the two acceptable values “Solution 1” and “Solution 2”. If the value is “Solution 1”, s^* remains unchanged. Otherwise, if the value is “Solution 2”, s^* is reassigned with s_i . If the answer does not fit the required format or value, the answer is considered invalid. In this situation, a simple strategy is to keep s^* unchanged. Similarly to the filtering task, complex strategies to re-prompt the LLM can be tested but are not covered in this work. There are four possible cases: the policy is not applicable to any of the solutions; the first solution is more complying with the policy than the second solution; the second solution is more complying with the policy than the first solution; both solutions comply equally with the policy. In both the first and final case, picking any of the two solutions is deemed correct. Once all solutions in $\bar{S}$ have been compared to s^* , the resulting s^* is considered the best solution and its index is returned.

In the event of failure due to too strict policies, LLM failure, or time constraints, the system can fall in a degraded mode of operation, in which a solution from the solver is chosen using classical optimization methods.

IV. EXPERIMENTAL EVALUATION

A. Setup

For the experimental evaluation the air traffic control simulator BlueSky, an open-source research-oriented platform developed by the Technical University of Delft, was used [43]. BlueSky provides a flexible and extensible environment for simulating realistic air traffic scenarios, enabling the testing of various automation and control strategies under controlled conditions. The simulator supports customizable scenario settings, sector configurations, and conflict detection algorithms. Its modular python-based architecture allows seamless integration of AI-based components such as natural language interfaces, making it well-suited for testing novel concepts of utilizing AI-based systems in air traffic control.

In the experimental scenario, two aircraft were introduced into the simulated airspace on intersecting trajectories, intentionally placing them on a collision course to trigger a conflict detection event. After the conflict is detected by the system, a CR solver generates a set of solutions. The set of solutions is filtered and sorted, as described in Subsection III-B. The commands of the best solution are scheduled into the BlueSky commands stack.

Regarding the LLM experimental setup, the open-source library `ollama`³ was used which facilitates the accessibility and reproducibility of the experiments, is easily integrated into BlueSky, and supports a wide range of state-of-the-art large language models. In line with the open-source nature of

³<https://github.com/ollama/ollama>

the software tools used and mentioned above the algorithm is tested with the Llama 3.1 [44] open-source language model series. Llama 3.1 is natively supported within the ollama library and is available in three sizes, specifically of 8b, 70b, and 405b parameters. These two features contributed to the choice of model, as well as its state-of-the-art performance in reasoning tasks. Experiments with the largest model were not executed due to the amount of resources required, making it unlikely to be used on classical hardware available in ATC facilities. The full source code of the setup including the BlueSky module and ollama integration can be accessed on GitHub⁴.

B. Method

1) *Dataset*: The performance of the filtering and sorting methods has been evaluated on small custom synthetic datasets. The filtering dataset is composed of 6 different sets of policies. Two of these sets contain only one policy, and 4 of these sets contain 2 policies (see example in Section A). Each of the 6 sets of policies is tested against 500 unique solutions, making 3000 entries. Each entry is composed of a solution, a filtering policy, and a Boolean indicating if the solution violates the policy. The policy violation is annotated using human-coded rules for each policy. The set of unique solutions is composed of 500 solutions, with 1 to 5 command(s) (equally distributed) per solution. The violation distribution is 72 % *yes* and 28 % *no*.

The sorting dataset is composed of seven different sets of policies. All policy sets contain only one policy (see example in Section A). Each of the 7 sets of policies is tested against 200 unique solutions, making 1400 entries. Each entry is composed of two solutions to compare, one policy set, and which solution is the best. The latter has four possible values (cf. Subsection III-B), distributed as such *not applicable* (33 %), *solution 1 preferred* (23 %), *solution 2 preferred* (23 %), *both solutions are good* (21 %).

2) *Decoding*: Different decoding strategies have been tested, namely *greedy* search decoding and *top-p sampling* [45]. In greedy decoding, no randomness is used. Models always generate the next most probable token based on the previous ones. Top-p decoding introduces randomness by picking a random token in the pool of the most probable tokens for which the sum of the probability of the pool is higher than *top_p*. The randomness is increased by increasing the *temperature* and *top_p* parameters [45].

The decoding parameters used in the experiments are, according to the ollama API option convention: *greedy* parameters: {'temperature': -1}, *low temperature* parameters: {'seed': 1, 'temperature': 0.5, 'top_p': 0.5}, *high temperature* parameters: {'seed': 1, 'temperature': 0.9, 'top_p': 0.95}.

C. Results

1) *Filtering task*: The best results for the filtering tasks were obtained with the model llama3.1:70b, using greedy decoding (cf. Table I). The bigger model always performed better than the smaller one, regardless of the decoding parameters.

2) *Sorting task*: The best results were obtained with llama3.1:70b, regardless of the decoding parameters (cf. Table II). It is important to note that due to the nature of the task and the dataset, all entries for which the policy is *not*

Table I: Results of the filtering task using different models.

Model	Parameters	Success Rate (%)
llama3.1:8b	greedy	80
llama3.1:8b	low temperature	81
llama3.1:8b	high temperature	81
llama3.1:70b	greedy	87
llama3.1:70b	low temperature	86
llama3.1:70b	high temperature	85

Table II: Results of the sorting task using different models.

Model	Parameters	Success Rate (%)
llama3.1:8b	greedy	90
llama3.1:8b	low temperature	90
llama3.1:8b	high temperature	88
llama3.1:70b	greedy	99
llama3.1:70b	low temperature	99
llama3.1:70b	high temperature	99

applicable or *both solutions are good* are always successful, as both solutions are considered valid (cf. Subsection III-B). These two cases represented 54 % of the entries (cf. Subsubsection IV-B1).

D. Analysis

1) *Success Rate*: All success rates for filtering were significantly higher than 50 %, excluding random guesses of the answer. All success rates were above 80 % supporting the hypothesis that the LLM understood the task and performed correct reasoning in the majority of the cases. However, the best success rate for filtering was 87 %, which is low considering the simplicity of the task which uses very simple policies and a maximum of two policies. Better performances were expected since llama3.1:70b performs with similar accuracy on more complex reasoning benchmarks [44], [46]. This low performance could be due to the lack of ATC-specific knowledge or the extensive length of the prompts, making it difficult for small/medium models to identify the correct elements of the prompt relevant to answering. Another explanation could be the failure of the model to perform correct reasoning.

All success rates for sorting are significantly higher than 77 %, excluding random guesses of the answer between case 2 and case 3 (cf. Subsection IV-C, Subsection III-B). The best success rate is 99 % which is satisfactory for sorting based on a policy. It is expected that this success rate would drop drastically with more than one policy because of the increase of complexity and ambiguity of combining sorting policies, even for humans.

2) *Failure cases*: The failure cases in the experiments exhibit, except for some exceptions, patterns related to hallucinations, misinterpretation of rules, or misalignment to the provided explanations of the language model (cf. Section A for examples of explanations provided by the LLM).

In the case of the filtering task the majority of the failures were false negatives—classified as not violating the given policy. Rules were frequently falsely referenced in the LLMs' explanation as accepting instead of forbidding a command. Moreover, in many cases the answer was "no" (violation) while the respective explanation described the existence of a violation. While the LLM showed a problematic language misunderstanding it is important to consider that the vast majority of the failures stemmed from one type of core rule,

⁴<https://github.com/DLR-KI/ATC-LLM-Interface>

namely “F1: Do not use a command with either a change of altitude nor speed.” (89 % and 44 % for the 70b and 8b model respectively). These examples highlight weaknesses of the model to reason with specific formulations, such as the use of negation in logic, or complex logic patterns like “nor”. It is worth noting that these examples are also non-trivial even for humans, as the amount of ambiguity—if measurable—is higher than for affirmative sentences and simple logic operators. Based on these observations, it would be valuable to explore questions such as “How is ambiguity in a task measured?” and “How should we deal with an ambiguous task?”. Further investigations would allow us to derive requirements on the tasks-space LLMs are qualified to operate in. In other cases, the explanation showed correct reasoning but a wrong answer. This highlights a limit of the basic prompting methods used in this prototype to induce an answer based on a reasoning step. Robust reasoning with LLM is an active part of research [37], [42], [47], and future work on implementing advanced reasoning techniques could highly improve and safeguard reasoning.

The sorting task failed in the majority of cases with negating rules (93 % and 63 % for the 70b and 8b models respectively) such as “G1: Prefer a solution which does not involve a change of heading.”. Furthermore, the sorting task was not understood as the LLMs’ explanations repeated the core rules but still returned the less preferred solutions as the final answer. One example generated by the 70b model was: “This answer follows core rule G1, which states to prefer a solution that does not involve a change of heading. Solution 1 only changes the heading once at the end, whereas Solution 2 does not change the heading at all but instead involves multiple altitude changes.” Notably, the LLM wrongly chose the first solution which contradicts the rule of not using heading commands. The misalignment to the policies was also visible in generated explanations which, instead of focusing on the given policy, revolved around preferring solutions with fewer commands, while this preference was never specified. Finally, especially the 70b model showed weak reasoning while providing lengthy and false justifications for it (i.e. hallucinations). Similarly to the analysis in the previous paragraph, the sorting task requires further work on reasoning safeguarding.

Both tasks revealed significant misinterpretation of the ATC-specific policies showing a lack of domain-specific knowledge. Possible solutions to remediate the lack of domain-specific knowledge includes fine-tuning a model with ATC-specific data, enriching the context with *Retrieval-Augmented Generation* [48], or incorporating experts’ knowledge with *reinforcement learning from human feedback* [49].

V. CONCLUSION AND OUTLOOK

This work aims to improve the alignment between ATCOs and conflict resolution solvers in the setting of SCOs during en-route ATC. The prototype described and tested in Section III and Section IV offers a proof of concept for the filtering and sorting of solutions provided by CR solvers in alignment with natural language policies. Such a system can allow ATCOs to dynamically specify preferences to solve conflicts without being restricted to predetermined optimization functions from classical CR solvers. The experiments in Section IV, describing the testing of the prototype on a custom synthetic dataset, validate that LLMs can perform reasoning on ATC tasks by successfully solving the majority of filtering and

sorting of CR solutions according to very simple policies. Despite the overall success, several limitations of the LLM are identified through the analysis of failure cases.

The experimentation with the widely applied Llama3.1 model series highlighted the existing challenges in natural language understanding in the presence of clear and analytically described reasoning tasks. These challenges were difficult to circumvent even with advanced prompting mechanisms. Although stronger reasoning models exist, this setup which opts for reproducibility, effortless installation and experimentation directs the application of LLMs to safety critical domains without compromising safety, supporting dynamic interaction and control between human operators and digital tools.

This proof of concept can be used to gather experience on LLMs and more generally natural language processing as the interface between ATCOs and ATC tools. Valuable experiences would be to identify the pitfalls of such a system and develop safeguarding mechanisms. Future work could contribute to safeguarding the system by answering the following open questions: “How to identify ambiguity in the task?”, “When identified, how to deal with ambiguous tasks?”, “How to safeguard reasoning?”, and “How to incorporate ATC knowledge in LLMs?”. Additionally, this prototype can be used to conduct experiments with ATCOs to gather first experiences and derive requirements from system users. Relevant questions to guide these experiments include: “What is expected from the system in case of failure to follow the policies?”, “What information should be part of the interaction with the system, and in which medium (e.g. visual, vocal) should it be conveyed?”

REFERENCES

- [1] DFS Deutsche Flugsicherung GmbH, “Luftverkehr in Deutschland. Mobilitätsbericht 2023,” 2023. [Online]. Available: <https://www.dfs.de/homepage/de/medien/publikationen/>
- [2] K. Langford, T. Kille, S.-Y. Lee, Y. Zhang, and P. R. Bates, ““in automation we trust” - australian air traffic controller perspectives of increasing automation in air traffic management,” *Transport Policy*, vol. 125, pp. 352–362, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0967070X22001822>
- [3] C. L. Tafur, R. G. Camero, D. A. Rodríguez, J. C. D. Rincón, and E. R. Saenz, “Applications of artificial intelligence in air operations: A systematic review,” *Results in Engineering*, vol. 25, p. 103742, 2025. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2590123024019856>
- [4] S. Malakis, M. Baumgartner, N. Berzina, T. Laursen, A. Smoker, A. Poti, G. Fabris, S. Velotto, M. Scala, and T. Kontogiannis, “A framework for supporting adaptive human-ai teaming in air traffic control,” in *Engineering Psychology and Cognitive Ergonomics*, D. Harris and W.-C. Li, Eds. Cham: Springer Nature Switzerland, 2023, pp. 320–330.
- [5] EASA, *EASA Artificial Intelligence Concept Paper Issue 2: Guidance for Level 1 & 2 machine-learning applications*. EASA, 2024. [Online]. Available: easa.europa.eu/ai
- [6] D. Yang, “Human-ai interaction in the age of large language models,” *Proceedings of the AAAI Symposium Series*, vol. 3, no. 1, pp. 66–67, 2024.
- [7] I. Gerdes, M. Jameel, R. Hunger, and Christoffels, “The automation evolves: Concept for a highly automated controller working position,” *33rd Congress of the International Council of the Aeronautical Sciences ICAS*, 2022. [Online]. Available: https://www.icas.org/icas_archive/ICAS2022/data/preview/ICAS2022_0268.htm
- [8] M. Jameel, L. Tyburzy, I. Gerdes, A. Pick, R. Hunger, and L. Christoffels, “Enabling digital air traffic controller assistant through human-autonomy teaming design,” in *2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC)*, 2023, pp. 1–9.
- [9] SKYbrary, “Conflict detection,” <https://skybrary.aero/articles/conflict-detection>, Year Published: 2021 / Last Updated: 2024, accessed: July 2024.
- [10] J. Kuchar and L. Yang, “A review of conflict detection and resolution modeling methods,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 4, pp. 179–189, 2000.

- [11] M. Prandini, J. Hu, J. Lygeros, and S. Sastry, "A probabilistic approach to aircraft conflict detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 4, pp. 199–220, 2000.
- [12] Y. Z. Siqi HAO, Shaowu CHENG, "A multi-aircraft conflict detection and resolution method for 4-dimensional trajectory-based operation," *Chinese Journal of Aeronautics*, vol. 31(7), pp. 1579–1593, 2018, available online 15 May 2018.
- [13] L. Pallottino, E. Feron, and A. Bicchi, "Conflict resolution problems for air traffic management systems solved with mixed integer programming," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 3, pp. 3 – 11, 04 2002.
- [14] F. H. Dias, H. Hijazi, and D. Rey, "Disjunctive linear separation conditions and mixed-integer formulations for aircraft conflict resolution," *European Journal of Operational Research*, vol. 296, no. 2, pp. 520–538, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0377221721002897>
- [15] M. Ribeiro, J. Ellerbroek, and J. Hoekstra, "Distributed conflict resolution at high traffic densities with reinforcement learning," *Aerospace*, vol. 9, no. 9, 2022. [Online]. Available: <https://www.mdpi.com/2226-4310/9/9/472>
- [16] International Civil Aviation Organization, *International Standards and Recommended Practices: Annex 11 - Air Traffic Services*, 14th ed. ICAO, 2016.
- [17] European Organisation for the Safety of Air Navigation, "ATM Cost-Effectiveness (ACE) 2019 Benchmarking Report with Special Focus on COVID-19 Impacts in 2020," p. 16, 2021. [Online]. Available: [https://www.eurocontrol.int/sites/default/files/2021-05/eurocontrol-ace-202019-benchmarking-report-special-focus-covid-19-impact-2020.pdf](https://www.eurocontrol.int/sites/default/files/2021-05/eurocontrol-ace-2019-benchmarking-report-special-focus-covid-19-impact-2020.pdf)
- [18] European Organisation for the Safety of Air Navigation, "Network Manager: Annual Report 2019," 2020. [Online]. Available: <https://www.eurocontrol.int/publication/network-manager-annual-report-2019>
- [19] European Organisation for the Safety of Air Navigation, "Network Manager: Annual Report 2021," p. 6, 2022. [Online]. Available: <https://www.eurocontrol.int/publication/network-manager-annual-report-2021>
- [20] R. Hunger, L. Christoffels, M. Friedrich, M. Jameel, A. Pick, I. Gerdes, P. M. von der Nahmer, and F. Sobotzki, "Lesson learned: Design and perception of single controller operations support tools," in *Engineering Psychology and Cognitive Ergonomics*, D. Harris and W.-C. Li, Eds. Cham: Springer Nature Switzerland, 2024, pp. 15–33.
- [21] M. Heinrich, *Moderne Flugsicherung*. Springer Vieweg, 2014.
- [22] A. R. Bestugin, E. A. A. F. A. D. P. A. P. S. A. Y. and S. Y. G, *Air Traffic Control Automated Systems*. Springer, 2020.
- [23] E. Haugg, A. Herber, and S. Herr, "Vorhaben cato – betriebliche nutzbarkeit von controller-tools für den unteren luftraum," *TE im Fokus*, vol. 2, pp. 11–18, 2010.
- [24] T. Stefani, F. Deligiannaki, C. Berro, M. Jameel, R. Hunger, C. Bruder, and T. Krüger, "Applying the assessment list for trustworthy artificial intelligence on the development of ai supported air traffic controller operations," in *2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC)*, C. Folkerts, Ed., vol. 2. IEEE, 2023, pp. 1–9. [Online]. Available: <https://elib.dlr.de/197961/>
- [25] T. Stefani, M. Jameel, I. Gerdes, R. Hunger, C. Bruder, E. Hoemann, J. M. Christensen, A. A. Giriya, F. Köster, T. Krüger, and S. Hallerbach, "Towards an operational design domain for safe human-ai teaming in the field of ai-based air traffic controller operations," in *2024 AIAA DATC/IEEE 43rd Digital Avionics Systems Conference (DASC)*. IEEE, 2024, pp. 1–10. [Online]. Available: <https://ieeexplore.ieee.org/document/10749445>
- [26] C. Bruder, S. Theis, T. Stefani, M. Jameel, D. Schulze Kissing, and D. German, "User participation in the design of trustworthy human-ai-collaboration in air-traffic control," 06 2024.
- [27] Bruder, Carmen und Theis, Sabine und Stefani, Thomas und Jameel, Mohsan und Hunger, Robert und Schulze Kissing, Dirk, "User participation in the design of trustworthy human-ai-collaboration in air-traffic control," Luebeck, Germany, 16.04.2024. [Online]. Available: https://www.researchgate.net/publication/381489821_USER_PARTICIPATION_IN_THE_DESIGN_OF_TRUSTWORTHY_HUMAN-AI-COLLABORATION_IN_AIR-TRAFFIC_CONTROL
- [28] Y. Liu, "Large language models for air transportation: A critical review," *Journal of the Air Transport Research Society*, vol. 2, p. 100024, 2024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2941198X24000356>
- [29] K. L. Fox, K. R. Niewoehner, M. Rahmes, J. Wong, and R. Razdan, "Leverage large language models for enhanced aviation safety," in *2024 Integrated Communications, Navigation and Surveillance Conference (ICNS)*, 2024, pp. 1–11.
- [30] L. Tyburzy, M. Jameel, R. Hunger, and J. Böhm, "Empowering human-ai collaboration in air traffic control through smart interaction design," in *2024 AIAA DATC/IEEE 43rd Digital Avionics Systems Conference (DASC)*, 2024, pp. 1–9.
- [31] S. Theis, S. Jentzsch, F. Deligiannaki, C. Berro, A. P. Raulf, and C. Bruder, "Requirements for explainability and acceptance of artificial intelligence in collaborative work," in *Artificial Intelligence in HCI*, H. Degen and S. Ntoa, Eds. Cham: Springer Nature Switzerland, 2023, pp. 355–380.
- [32] D. Guo, Z. Zhang, P. Fan, J. Zhang, and B. Yang, "A context-aware language model to improve the speech recognition in air traffic control," *Aerospace*, vol. 8, no. 11, 2021. [Online]. Available: <https://www.mdpi.com/2226-4310/8/11/348>
- [33] X. Wang, H. Ding, Q. Xu, and Z. Liu, "Information extraction of air-traffic control instructions via pre-trained models," in *Artificial Intelligence in China*, Q. Liang, W. Wang, J. Mu, X. Liu, and Z. Na, Eds. Singapore: Springer Singapore, 2022, pp. 82–91.
- [34] U. Singh, M. Bhattacharya, and R. Padhi, "State-of-the-art natural language processing for aviation: A review," *TechRxiv*, Mar. 2025. [Online]. Available: <http://dx.doi.org/10.36227/techrxiv.174198516.68024838/v1>
- [35] S. Abdulhak, W. Hubbard, K. Gopalakrishnan, and M. Z. Li, "Chatatc: Large language model-driven conversational agents for supporting strategic air traffic flow management," 2024. [Online]. Available: <https://arxiv.org/abs/2402.14850>
- [36] J. Andriuskevičius and J. Sun, "Automatic control with human-like reasoning: Exploring language model embodied air traffic agents," 2024. [Online]. Available: <https://arxiv.org/abs/2409.09717>
- [37] Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon, C. Chen, C. Olsson, C. Olah, D. Hernandez, D. Drain, D. Ganguli, D. Li, E. Tran-Johnson, E. Perez, J. Kerr, J. Mueller, J. Ladish, J. Landau, K. Ndousse, K. Lukosuite, L. Lovitt, M. Sellitto, N. Elhage, N. Schiefer, N. Mercado, N. DasSarma, R. Lasenby, R. Larson, S. Ringer, S. Johnston, S. Kravec, S. E. Showk, S. Fort, T. Lanham, T. Telleen-Lawton, T. Conerly, T. Henighan, T. Hume, S. R. Bowman, Z. Hatfield-Dodds, B. Mann, D. Amodei, N. Joseph, S. McCandlish, T. Brown, and J. Kaplan, "Constitutional ai: Harmlessness from ai feedback," 2022. [Online]. Available: <https://arxiv.org/abs/2212.08073>
- [38] M. Ahn, D. Dwibedi, C. Finn, M. G. Arenas, K. Gopalakrishnan, K. Hausman, B. Ichter, A. Irpan, N. Joshi, R. Julian, S. Kirmani, I. Leal, E. Lee, S. Levine, Y. Lu, I. Leal, S. Maddineni, K. Rao, D. Sadigh, P. Sanketi, P. Sermanet, Q. Vuong, S. Welker, F. Xia, T. Xiao, P. Xu, S. Xu, and Z. Xu, "Autort: Embodied foundation models for large scale orchestration of robotic agents," 2024. [Online]. Available: <https://arxiv.org/abs/2401.12963>
- [39] Z. Xu, S. Jain, and M. Kankanalli, "Hallucination is inevitable: An innate limitation of large language models," 2025. [Online]. Available: <https://arxiv.org/abs/2401.11817>
- [40] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung, "Survey of hallucination in natural language generation," *ACM Comput. Surv.*, vol. 55, no. 12, Mar. 2023. [Online]. Available: <https://doi.org/10.1145/3571730>
- [41] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," 2023. [Online]. Available: <https://arxiv.org/abs/2201.11903>
- [42] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," 2023. [Online]. Available: <https://arxiv.org/abs/2210.03629>
- [43] J. Hoekstra and J. Ellerbroek, "Bluesky atc simulator project: an open data and open source approach," in *International Conference for Research on Air Transportation, Volume: 7*, 06 2016.
- [44] Llama team at Meta AI, "The llama 3 herd of models," 2024. [Online]. Available: <https://arxiv.org/abs/2407.21783>
- [45] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," 2020. [Online]. Available: <https://arxiv.org/abs/1904.09751>
- [46] "Llama 3.1 with multilinguality and long context — huggingface.co," <https://huggingface.co/blog/llama31>, 2024, [Accessed 13-01-2025].
- [47] F. Xu, Q. Hao, Z. Zong, J. Wang, Y. Zhang, J. Wang, X. Lan, J. Gong, T. Ouyang, F. Meng, C. Shao, Y. Yan, Q. Yang, Y. Song, S. Ren, X. Hu, Y. Li, J. Feng, C. Gao, and Y. Li, "Towards large reasoning models: A survey of reinforced reasoning with large language models," 2025. [Online]. Available: <https://arxiv.org/abs/2501.09686>
- [48] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 9459–9474. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf
- [49] Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan *et al.*, "Training a helpful and harmless assistant with reinforcement learning from human feedback," *arXiv preprint arXiv:2204.05862*, 2022.

APPENDIX

A. Filtering prompt

```
You are an assistant to an Air Traffic Controller.

Do not guess. Only use the facts.

# Facts
- The heading command changes the direction of the
  aircraft. heading does not modify the speed nor
  the altitude.
- The climb command changes the altitude of the
  aircraft. climb does not modify the speed nor the
  direction.
- The speed command changes the speed of the aircraft.
  speed does not modify the altitude nor the
  direction.

# Core rules
F1: Do not use a command involving a change of speed.
F2: Do not use a command involving a change of
  altitude.
You need to make sure that no core rule is violated by
the following solution.

# Solution
```json
[
 {time: 4:28:00, value: 188, command: Change
 heading to 188deg}
]
```

# Answer format
```json
{
 Explanation: [One sentence explanation based on
 rules and solution],
 Answer: [yes/no]
}
```

# Inquiry
Does the provided solution violate one of the core
rules?

Answer the inquiry in JSON in code block.
```

LLM answer:

```
```json
{
 "Explanation": "The solution only changes the
 direction of the aircraft, which does not involve
 a change of speed or altitude.",
 "Answer": "no"
}
```
```

B. Sorting prompt

```
You are an assistant to an Air Traffic Controller.

Do not guess. Only use the facts.

# Facts
- The heading command changes the direction of the
  aircraft. heading does not modify the speed nor
  the altitude.
- The climb command changes the altitude of the
  aircraft. climb does not modify the speed nor the
  direction.
- The speed command changes the speed of the aircraft.
  speed does not modify the altitude nor the
  direction.

# Core rules
G1: Prefer a solution which does not involve a change
  of altitude.

You will encounter some core rules that you need to
always satisfy when choosing the correct
solutions for the
ATCO.
The rules express preference over different options.

The order of the rules does matter.

Always try to satisfy the first rule. If the first
rule is not applicable, use the second.

# Solution 1
```json
[
 {time: 1:03:23, value: 325, command: Change
 heading to 325deg}
]
```

# Solution 2
```json
[
 {time: 11:13:06, value: 181, command: Change
 heading to 181deg},
 {time: 14:49:28, value: 6600, command: Change
 altitude to 6600m},
 {time: 22:50:48, value: 182, command: Change speed
 to 182m/s}
]
```

# Accepted answer format
```json
{
 Explanation: [One sentence explanation based on
 rules],
 Answer: [Solution 1/Solution 2]
}
```

It is required to always provide a definite answer
(solution 1 or solution 2) otherwise the program
fails.

If both solutions are equal with respect to the core
rules, any of the two solutions can be chosen.

# Inquiry
Based on the core rules, which solution is preferred?

Answer the inquiry in JSON in code block.
```

LLM answer:

```
```json
{
 Explanation: Solution 1 is preferred as it does
 not involve a change of altitude, satisfying the
 first core rule (G1).,
 Answer: Solution 1
}
```
```

C. Filtering dataset sample

| Solution | Policy | Violation |
|--|--|-----------|
| <pre>{ "callsign": "CXK471", "commands": [{ "time": "5:16:50", "value": 1200, "command": "Change altitude to 1200m" }, { "time": "19:59:33", "value": 7800, "command": "Change altitude to 7800m" }] }</pre> | <pre>{ "F1": "Do not use a command involving a change of speed.", "F2": "Do not use a command involving a change of altitude." }</pre> | True |
| <pre>{ "callsign": "ZYZ896", "commands": [{ "time": "12:22:31", "value": 132, "command": "Change heading to 132deg" }, { "time": "0:41:41", "value": 35, "command": "Change heading to 35deg" }] }</pre> | <pre>{ "F1": "Do not use a command with either a change of altitude nor speed." }</pre> | False |

D. Sorting dataset sample

| Solution_1 | Solution_2 | Policy | Case |
|--|---|---|------|
| <pre>{ "callsign": "UDA293", "commands": [{ "time": "21:02:18", "value": 327, "command": "Change heading to 327deg" }] }</pre> | <pre>{ "callsign": "QRX507", "commands": [{ "time": "22:16:42", "value": 6600, "command": "Change altitude to 6600m" }] }</pre> | <pre>{ "G1": "Prefer a solution which does not involve a change of altitude." }</pre> | 1 |
| <pre>{ "callsign": "ZUL206", "commands": [{ "time": "15:07:07", "value": 257, "command": "Change speed to 257m/s" }, { "time": "15:57:17", "value": 260, "command": "Change speed to 260m/s" }, { "time": "20:28:17", "value": 254, "command": "Change speed to 254m/s" }] }</pre> | <pre>{ "callsign": "SRJ237", "commands": [{ "time": "4:35:42", "value": 9000, "command": "Change altitude to 9000m" }] }</pre> | <pre>{ "G1": "Prefer a solution which does not involve a change of speed." }</pre> | 2 |