Introduction

The gatekeeper architecture for AI systems proposed in this programme is a promising architecture to build safer machine learning-based systems [19]. It is designed to improve the trustworhiness and safety of an AI system by limiting the interactions of machine learnt modules with the external world to interactions that are approved by a *verifier* module. The verifier module's design and implementation is assumed to have the highest levels of trustworthiness and reliability. Our **vision** is that the verifier could best achieve this standard through *formally mathematically* proving that it fulfils desired trustworthiness and reliability properties and that this could be most feasibly achieved using an interactive theorem prover (ITP) with a minimal trusted code base.

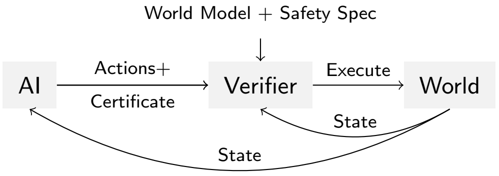


Figure 1: *A gatekeeper architecture to improve the safety and trustworthiness of machine learning-based systems [19].*

The main objective of TA1.1 is the development of a modelling language that could capture rich descriptions of world dynamics, including continuous change, randomness, and non-determinism, and also safety specifications. This language is sought to enable the verifier to communicate with the machine learnt module (henceforth, the AI) to verify that its output can be let through, and also with human auditors who make sure that the verifier has an accurate world-model and safety specifications against which to check the AI's output. In order for the verifier to be designed and implemented with formal mathematical guarantees, it is imperative that the developed language has *formal semantics* specified (aka formalised) in an ITP's logic and that the verifier module is proven to correct w.r.t. those formal semantics within the ITP. We envisage **challenges** to achieving that. Those challenges include

1. building powerful enough formal mathematical libraries to enable the formal definition of the concepts involved in the language, like ODEs, PDEs, MDPs, etc.,
2. having logical/category-theoretic constructs required to formalise the semantics of the language feasibly, elegantly, and compositionally in the ITP's logic prover in a way that is understandable to human auditors,
3. the feasibility of designing and formally verifying certificate checking algorithms for the language, and
4. methodological challenges relating to having the certificate checkers be (efficiently) executable.

To make those challenges concrete, take a concept like MDPs, on the formalisation of which we have already made significant progress [44], and which is cornerstone for any language capable of modelling randomness. Formalising the semantics of MDPs requires a lot of deep formal mathematical concepts and results, like measure theory, probabilities, and limits. Furthermore, the semantics of MDPs need advanced logical/category-theoretic constructs, like the Giry monad [25, 20], which in turn needs co-recursion [47] for its definition, all of which are advanced concepts that are non-trivial to build in an ITP. Lastly, formally verifying algorithms that operate on MDPs needs the development of verification methodologies for approximating computation on reals, like floating point computation, which is a difficult problem that is currently open.

We believe that the design of the language should be informed early-on in the design process by formalising its semantics in an ITP's logic. This is to avoid unmanageable technical debt caused by those four challenges if, at a late stage of the design process, the language's semantics are to be formalised in an ITP or if the verifier is to be formally proved correct w.r.t. the semantics. To do that, we will pursue the following two **objectives** if this proposal is successful.

1. We will formalise, in an ITP's logic, the semantics of action languages that include (combinations) of constructs that will inevitably be needed to arrive at the language targeted by TA1.1. This should include constructs to model world dynamics and to specify verification conditions.
2. We will design and formally verify using an ITP, w.r.t. the formal semantics of the language, validators and certificate checkers for plans and policies.

Background

Planning Domain Definition Language (PDDL) [36] is the de facto standard language in which the planning community conducts its research. International Planning Competitions have used PDDL since 1998 and most planning systems accept PDDL as input. In PDDL, a planning problem is defined as a *domain*, which specifies schemata of predicates and actions, and an *instance* of that domain, which specifies a set of objects, an initial state specifying what predicates apply to what objects initially, and a goal formula representing a logical condition that should hold in the goal state. In its simplest form, a solution is a sequence of actions, each of which applied to some of the objects in the instance, which transforms the initial state into a state satisfying the goal. Initially, PDDL was only capable of modelling classical planning problems [36], where action effects are discrete and deterministic, and where there is a finite number of possible states. Later that was extended to PDDL2.1 [23], which can model a dense timeline as well as continuous change. Another important extension is PDDL+ [22], which is capable of expressing hybrid planning problems, including exogenous events and processes. In addition to PDDL, we are also interested in RDDL [43], which is a language used to represent probabilistic planning problems, i.e. planning problems where the actions and states are modelled to involve randomness and where the underlying formalism is a factored MDP.

One main goal of the modelling language aimed for by the end of TA1.1 is that it would be able to incorporate a neural network or another machine learnt module as the AI whose output is to be verified before being executed. In our context, where the language is a planning language like PDDL or RDDL, we envision a neural network to be a policy for recommending planning actions, and those actions are to be *validated* by the verifier before being executed. This would involve making sure that 1. the actions can execute in the given order, 2. the actions achieve the goal, if executed from a specific state, and 3. the actions do not violate given safety specifications if executed. The first two items are already achievable for large fragments of planning languages using so-called *plan validators* [32, 27, 9, 7]. The third item has been studied in planning and model checking, where variants of temporal logic have been used to specify the safety specifications of the plans in a wide variety of contexts, e.g. deterministic [18, 24], probabilistic [16], with a timeline [15], and with continuous change [28].

The other goal of the language we focus on here is that it should allow humans to build, or at least understand, the world-model and the safety specifications under which the verifier operates. Again, planning languages like PDDL and RDDL can achieve that, as witnessed by the large number of human-built domains in these languages, e.g. planning competition domains, starting in 1998 [37].

Interactive Theorem Provers (ITPs) In this project we will use the ITP Isabelle/HOL [41], a theorem prover based on Higher-Order Logic, to formalise the semantics of the modelling language as well as to formally mathematically prove properties of the verifier. It is designed for trustworthiness: following the LCF approach [38], a small trusted kernel implements the inference rules of the logic, and, using encapsulation features of the functional programming language Standard ML, it guarantees that all theorems are actually proved by this small kernel. Around the kernel, there is a large set of tools that implement proof tactics and high-level concepts like algebraic datatypes and recursive functions. Bugs in these tools cannot lead to inconsistent theorems being proved, but only to error messages when the kernel refuses a proof.

Although there are other ITPs (e.g. Coq, Lean, HOLlight) which we could use, we propose the use of Isabelle/HOL primarily due to the existence of many necessary formal mathematical libraries, developed previously by the PI and others, and also due to its extensive support for generating verified executable code in various languages [26]. Isabelle/HOL includes a rich, well-established formal mathematical libraries on analysis [31], including deep results from multi-variate analysis [13] and a verified state-of-the-art ODE solver [33], measure theory and probabilities [30, 29], and constructs to reason about probabilistic algorithms, like the Giry monad [20]. Most relevant to our project here, formal libraries in Isabelle/HOL include formalised semantics of the STRIPS and temporal planning, where there is the notion of dense timeline, fragments of PDDL by the PI and collaborators [9, 7] and a formalised model of MDPs with rewards along with algorithms to solve them, also by the PI and the PDRA [44]. In addition to this available material, Isabelle/HOL's trustworthiness can be significantly multiplied using a verified kernel [42].

```
definition "ranking ≡
  do {
    σ ← pmf-of-set (permutations-of-set V);
    return-pmf (online-match G π σ)
  }"
```

Figure 2: *A probablistic program modelled in Isabelle/HOL using the Giry monad. First uniform distribution on a set of permutations is sampled, then the PMF corresponding to the program is returned [10].*

Work Programme

Work Package 1: Formalising the Semantics of an Action Language

In the first work package (WP), we focus on objective 1 from above. We will formalise the semantics of a language capable of modelling discrete/continuous hybrid system dynamics, ordinary differential equations, concurrency, and (factored) MDPs, which are inevitable elements of the modelling language sought in TA1.1. Although we will not be able to model all the types of dynamics captured by the modelling languages in Appendix A of this programme's thesis, those four types of dynamics capture a lot of the semantic features needed to formalise the semantics of the rest. We will initially formalise semantics of languages used in the planning community, in particular PDDL+ [36, 23, 22], which includes hybrid change, a timeline, and concurrency, and RDDL [43], which mainly focuses on probabilistic dynamics. Then we will extend the semantics of those languages in two ways: a. to include (combinations of) features that are desirable for achieving the ambitious goals of TA1.1, but that are currently not supported by any existing planning languages, and b. to include extended specification languages to enable the specification of desirable verification properties, in addition to the planning-enteric existing specification languages. Example specifications would be in (probabilistic) linear temporal logic (LTL), where bounds on the probability of satisfying an LTL formula could be specified and proven. This will be done in Isabelle/HOL, which, as stated above, has a lot of the necessary background theories and infrastructure.

Task 1: Discrete/Continuous Hybrid Planning Semantics In this task we will formalise the semantics of actions which exhibit continuous change, i.e. actions whose effects are described as ODEs, which is a part of PDDL2.1 [23], as well as processes and exogenous events, which are a part of PDDL+ [22]. We will build on our existing formalisation of the classical [9] and temporal [7] fragments of PDDL, and on Immler's existing Isabelle/HOL formalisation of ODEs [33]. Thus, the semantics here will be mainly state-transition semantics directly encoded in Isabelle/HOL's logic, using the different existing recursion [35] and data type [47] facilities. In addition to defining the standard semantics of PDDL+, we will add to its semantics more involved temporal logical specifications on hybrid systems(e.g. from [28]) to specify more properties than the two basic properties supported by standard PDDL+ semantics: that the plan can indeed execute, i.e. action preconditions are always satisfied before executing them, and that by the end of executing the plan a goal state has been reached.

Task 2: Probabilistic Planning Semantics In this task we will formalise the semantics of probablistic planning problems. We will focus on the semantics of RDDL [43] due to the abundance of planning competition domains specified in this language from previous IPCs. Similar to the first task, we will perform the formalisation in Isabelle/HOL. Here we will build on our own

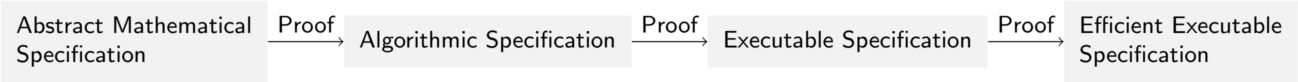

Abstract Mathematical Specification
Proof
Algorithmic Specification
Proof
Executable Specification
Proof
Efficient Executable Specification

Figure 3: *A sketch of the step-wise refinement process for developing formally verified software.*

formalisation of MDPs with rewards [44], based on which we will specify the semantics of RDDL as factored MDPs. More specifically, the semantics here will be built on the Giry monad, which allows for elegant modelling of randomised computation and recursion (see Fig 2), has reasonable practical support in Isabelle/HOL [20], and also has well-behaved category-theoretic formal semantics [25].

Task 3: Semantics of a Hybrid-Probabilistic Language A challenging task in this project is to extend the semantics of the languages formalised in the other tasks. Our main goal here is to formalise the semantics of a practically usable action language that allows for describing system dynamics that include randomness, discrete/continuous hybrid change, and concurrency. Currently, we are not aware of any language that practically combines those three features. There are a number of languages that support combinations of these languages. 1. RDDL, which has support for concurrency, but does not support continuous change, and 2. PPDDL, which supports continuous change, but a much more restrictive form of concurrency. The reasons for the above restrictions are more fundamental that just syntactic limitation. The semantics of RDDL are based on dynamic Bayesian networks, while PPDDL's semantics are an extension of state-transition semantics for PDDL. The difference in semantics makes it less straight forward to add or remove features to the languages. Here we aim at creating a practical language with compositional formal semantics, which has discrete/continuous hybrid change, and concurrency. One potential method for doing that is extending the Giry monad, which we will use for formalising the semantics of probabilistic planning. We will study following the same monadic approach, extending the Giry monad to allow for continuous change and concurrency, two features which have been studied before using monadic approaches in theorem provers, e.g. concurrent monads [17] and monads modelling continuous change [39].

WP 2: Formally Verified Validators and Certificate Checkers

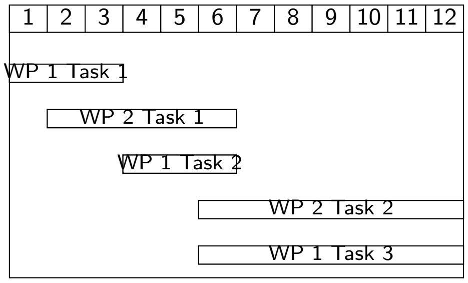
Formally verified validators and certificate checkers for action sequences or policies are in our perspective an essential part of building a practical AI with formal safety guarantees. This is of course tied to our choice of the modelling language as an action language, where the AI would generate actions in that action language and, before those actions are executed in the outside world, they are validated or certified to achieve the safety specifications. In this WP, we will construct prototype executable validators and certificate checkers verified w.r.t. the semantics of the modelling language. The methodology for constructing the verified validators will be based on step-wise refinement [48], sketched in Fig. 3. We start from the abstractly specified semantics and refine that specification towards an executable program which fulfils those abstractly specified semantics. This is done over multiple steps, where in every step more implementation details/optimisations are added to the specification, and it is formally proved that the new more detailed specification is equivalent to the less detailed one. That approach was used to verify most advanced algorithms which were ever verified [40] because it separates the verification effort related to abstract algorithmic aspects from the effort pertaining to implementation specifics and optimisations. In our context here, the goal of deriving executable validators and certificate checkers is to confirm that our semantics can be used to generate executable validator/certificate checkers, which is not always possible, either due to the semantics using uncomputable constructs, like real numbers or non-terminating recursion, or due to the difficulty of finding validation/certification algorithms for certain constructs in the modelling language.

Task 1: Validator for PDDL2.1 and PDDL+ In this task we will develop executable validators for full PDDL2.1 and PDDL+ plans. The implementation should be formally verified w.r.t. the semantics developed in WP 1 Task 1. This validator will build on the validator we developed in our earlier work for temporal planning problems [7], and its main goal will be to confirm that a given plan can execute, according to the semantics of action execution, and that it results in a goal state. We will use Isabelle/HOL's code generation to produce the implementation. The most challenging part of this task is specifying, implementing, and formally verifying the validator for continuous change, as we need a formally verified ODE solver. There is a verified ODE solver formalised within Isabelle/HOL [33] which we can use. However, given that this solver does not always terminate, we will also implement and verify a specialised solver akin to the one implemented by Fox and Long [21], which always terminates for interesting classes of problems, e.g. ones with only acyclic dependencies between the differential equations.

Task 2: Validator for Probabilistic Planning In this task we will construct an executable validator for solutions, a.k.a. policies, of probabilistic planning problems. The most basic property we would like to validate is that a given policy is an optimal policy, in terms of utility, for the given problem. We will focus on firstly designing a certificate checker for optimal policies. We plan to do that for two kinds of probabilistic planning problems: finite horizon probabilistic planning problems and infinite horizon problems. For both problems we will focus on certification based on certifying solutions of linear programs. We suspect we might either use a calculus-based approach, like Maribou [34], or an approach based on the primal-dual theorem and Farkas' lemma. Then we will verify a compilation of finite and infinite horizon planning to linear programs.

Further properties we will tackle here are probabilistic linear temporal logic (LTL) properties, e.g. that a given formula will be satisfied with a minimum or a maximum probability. Certifying such properties would be hard to achieve for all probabilistic planning algorithms, but we will be able to achieve that for specially modified planning algorithms that produces the needed certificate, e.g. using the approach of Sickert et al. [46].

Project Plan and Risk Management



The schedule of executing this project is shown above in terms of months from the project's beginning. The PI will be leading WP 1 and the PDRA will lead WP 2. Nonetheless, the work in both WP's will have to involve substantial collaboration between the PI and PDRA, since the design of the semantics and the different features will directly impact the possibility of devising validation or certification algorithms. Indeed, for the first month both the PI and the PDRA will draft the formal semantics of the hybrid planning language, before the PDRA starts working on validation or certification algorithms.

The primary risk comes from the dependencies of implementing validators and certificate checkers (i.e. WP 2 Task 1 and WP 2 Task 2) on formalising the semantics of the relevant fragments of PDDL (i.e. WP 1 Task 1 and WP 1 Task 2). This risk could manifest particularly if the PDRA does not have appropriate background in interactive theorem proving and/or if the PDRA is unable to deal with the steep learning curve of ITPs. We plan to avoid that risk by having the PI do the relatively shorter but riskier task of formalising the semantics in the theorem prover.

Team

The **PI**, Mohammad Abdulaziz, is a lecturer in Artificial Intelligence at King's College London and he will dedicate 60% of his time to the project throughout its entire duration. Before that he was a post-doctoral researcher at the Chair for Logic and Verification at TU Munich, which is the core team behind the development of the theorem prover Isabelle [41], and where he is still a guest researcher. Earlier, he was awarded a PhD from the Australian National University and an MSc from Cairo University.

The PI is one of the few researchers who are active in the application of interactive theorem provers to the verification of AI software and algorithms, making him one of the few people capable of delivering this proposed project. He has consistently published in top AI venues, like The AAAI Conference on Artificial Intelligence (AAAI), The International Joint Conference on Artificial Intelligence (IJCAI), and and The International Conference on Automated Planning and Scheduling (ICAPS), as well as top venues in the area of theorem proving, like The International Conference on Interactive Theorem Proving (ITP) and The Journal of Automated Reasoning (JAR).

The most related work by the PI and his collaborators to this proposal is the formalisation, in HOL, of the semantics of modelling languages for planning. This includes the classical [9] and temporal fragments [7] of PDDL, the main language used by the planning community. He also developed executable validators for solutions of planning problems modelled in the formalised fragments of PDDL. Those validators were verified w.r.t. the formalised semantics. Using those validators, multiple bugs were found in VAL [32], the main plan validator used by the planning community.

Furthermore, the PI, together with collaborators, has worked on formally verifying planing systems and algorithms. This includes a verified SAT-based planning system [8], verified solvers for infinite-horizon MDPs [44], verified symmetry breaking algorithms for planning problems [6], and verified algorithms to reason about state spaces [4, 1, 5, 12]. In addition to his work on applications of verification technology to AI planning, the PI is also active in formalising mathematics, with work ranging from the formalisation of results from mathematical analysis, like the formalisation of Green's theorem [14], to formalising deep results in theoretical computer science, like the correctness of matching [11, 10] and flow algorithms [3].

The PI's work on the application of theorem provers to the verification of AI systems and the formalisation of mathematics was recognised by being selected to speak in AAAI 2024's New Faculty Highlights programme [2], as well as being a keynote speaker in the International Conference on Graph Transformations 2023, the workshop on AI Secure by Construction in 2022, and the Hausdorff Institute of Mathematics School on Formal Mathematics in 2024. He is also regularly invited to the programme committees of top AI conferences, like AAAI, IJCAI, and ICAPS, as well as top conferences in interactive theorem proving, like the ACM Conference on Certified Programs and Proofs, and ITP.

In addition to the PI's research experience, he also has significant supervision experience. He is currently supervising three PhD students at King's College London, and (informally) one student at TU Munich, all working on applications of theorem provers to the formalisation of mathematics and the verification of algorithms.

The **post-doctoral research associate (PDRA)** we plan to hire, Maximilian Schäfeler, is currently a PhD student at the Chair of Logic and Verification at TU Munich, informally co-supervised by the PI. He will dedicate 100% of his time to the project, after he submits his thesis in August 2024. He has substantial experience in using Isabelle/HOL to formally reason about algorithms and mathematics. Until now, he has verified a large number of algorithms for reasoning about MDPs, ranging from value and policy iteration [44] to advanced algorithms for reasoning about factored MDPs [45]. This makes him a very fitting candidate for working on this project.

Why through ARIA It should be clear that the project proposed here is already a natural continuation of the team's current research. However, if this proposal is successful, there will be concrete advantages. First, with 60% of his time, the PI will be substantially more focused on this research agenda. For the PDRA, this project will provide him with extra funding to continue his research, given that he is about to finish his PhD funding. The second advantage is that the workshops and other meetups during the programme, proposed in the programme director's presentation, will provide an avenue for the PI and PDRA to exchange ideas with other researchers who are working towards complementary goals. This will provide us with better contextualisation for our work within a bigger community, all working towards safe AI, but from different angles. Our work will gain by this exchange of ideas, especially with other researchers whose expertise are far from ours.

References

[1] Mohammad Abdulaziz. Plan-length bounds: Beyond 1-way dependency. In *The 33rd AAAI Conference on Artificial Intelligence (AAAI)*, 2019. doi:10.1609/aaai.v33i01.33017502.

[2] Mohammad Abdulaziz. Interactive Theorem Provers: Applications in AI, Opportunities, and Challenges. *Proceedings of the AAAI Conference on Artificial Intelligence*, March 2024. doi:10.1609/aaai.v38i20.30276.

[3] Mohammad Abdulaziz and Thomas Ammer. A formal analysis of capacity scaling algorithms for minimum cost flows. In *To Appear in the 15th International Conference on Interactive Theorem Proving (ITP)*, 2024.

[4] Mohammad Abdulaziz and Dominik Berger. Computing plan-length bounds using lengths of longest paths. In *The 35th AAAI Conference on Artificial Intelligence (AAAI)*, 2021. doi:10.1609/aaai.v35i13.17392.

[5] Mohammad Abdulaziz, Charles Gretton, and Michael Norrish. A state space acyclicity property for exponentially tighter plan length bounds. In *The 27th International Conference on Automated Planning and Scheduling (ICAPS)*, 2017. doi:10.1609/icaps.v27i1.13837.

[6] Mohammad Abdulaziz, Charles Gretton, and Michael Norrish. A Verified Compositional Algorithm for AI Planning. In *The 10th International Conference on Interactive Theorem Proving (ITP)*, 2019. doi:10.4230/LIPIcs.ITP.2019.4.

[7] Mohammad Abdulaziz and Lukas Koller. Formal semantics and formally verified validation for temporal planning. In *The 36th AAAI Conference on Artificial Intelligence (AAAI)*, 2022. doi:10.1609/aaai.v36i9.21197.

[8] Mohammad Abdulaziz and Friedrich Kurz. Formally verified SAT-Based AI planning. In *The 37th AAAI Conference on Artificial Intelligence (AAAI)*, 2023. doi:10.1609/aaai.v37i12.26714.

[9] Mohammad Abdulaziz and Peter Lammich. A Formally Verified Validator for Classical Planning Problems and Solutions. In *The 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2018. doi:10.1109/ICTAI.2018.00079.

[10] Mohammad Abdulaziz and Christoph Madlener. A Formal Analysis of RANKING. In *The 14th Conference on Interactive Theorem Proving (ITP)*, 2023. doi:10.48550/arXiv.2302.13747.

[11] Mohammad Abdulaziz, Kurt Mehlhorn, and Tobias Nipkow. Trustworthy graph algorithms (invited paper). In *The 44th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, 2019. doi:10.4230/LIPIcs.MFCS.2019.1.

[12] Mohammad Abdulaziz, Michael Norrish, and Charles Gretton. Formally Verified Algorithms for Upper-Bounding State Space Diameters. *J. Autom. Reason.*, 2018. doi:10.1007/s10817-018-9450-z.

[13] Mohammad Abdulaziz and Lawrence C. Paulson. An Isabelle/HOL Formalisation of Green's Theorem. In *The 7th International Conference on Interactive Theorem Proving (ITP)*, 2016. doi:10.1007/978-3-319-43144-4_1.

[14] Mohammad Abdulaziz and Lawrence C. Paulson. An Isabelle/HOL Formalisation of Green's Theorem. *J. Autom. Reason.*, 2019. doi:10.1007/s10817-018-9495-z.

[15] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, April 1994. doi:10.1016/0304-3975(94)90010-8.

[16] Christel Baier, Edmund M. Clarke, Vasiliki Hartonas-Garmhausen, Marta Kwiatkowska, and Mark Ryan. Symbolic model checking for probabilistic processes. In *Automata, Languages and Programming*. 1997. doi:10.1007/3-540-63165-8_199.

[17] Koen Claessen. A poor man's concurrency monad. *J. Funct. Prog.*, May 1999. doi:10.1017/S0956796899003342.

[18] Edmund M. Clarke, Jr., Orna Grumberg, and Doron A. Peled. *Model Checking*. 1999.

[19] David "davidad" Dalrymple, Joar Skalse, Yoshua Bengio, Stuart Russell, Max Tegmark, Sanjit Seshia, Steve Omohundro, Christian Szegedy, Ben Goldhaber, Nora Ammann, Alessandro Abate, Joe Halpern, Clark Barrett, Ding Zhao, Tan Zhi-Xuan, Jeannette Wing, and Joshua Tenenbaum. Towards Guaranteed Safe AI: A Framework for Ensuring Robust and Reliable AI Systems, May 2024. URL: http://arxiv.org/abs/2405.06624, arXiv:2405.06624.

[20] Manuel Eberl, Johannes Hölzl, and Tobias Nipkow. A Verified Compiler for Probability Density Functions. In *Programming Languages and Systems - 24th European Symposium on Programming, ESOP 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings*, 2015. doi:10.1007/978-3-662-46669-8_4.

[21] Maria Fox, Richard Howey, and Derek Long. Validating Plans in the Context of Processes and Exogenous Events. In *20th National Conference on Artificial Intelligence (NCAI)*, 2005. URL: `http://www.aaai.org/Library/AAAI/2005/aaai05-182.php`.

[22] Maria Fox and Derek Long. PDDL+: Modeling continuous time dependent effects. In *The 3rd International NASA Workshop on Planning and Scheduling for Space*, 2002.

[23] Maria Fox and Derek Long. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *J. Artif. Intell. Res.*, 2003. URL: `http://arxiv.org/abs/1106.4561`.

[24] Alfonso Gerevini and Derek Long. Plan Constraints and Preferences in PDDL3.

[25] Michele Giry. A categorical approach to probability theory. In *Categorical Aspects of Topology and Analysis*, 1982. `doi:10.1007/BFb0092872`.

[26] Florian Haftmann and Tobias Nipkow. A code generator framework for Isabelle/HOL. Technical report, Department of Computer Science, University of Kaiserslautern, 2007.

[27] Patrik Haslum. Patrikhaslum/INVAL, January 2024. URL: `https://github.com/patrikhaslum/INVAL`.

[28] T.A. Henzinger. The theory of hybrid automata. In *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, July 1996. `doi:10.1109/LICS.1996.561342`.

[29] Johannes Hölzl. Markov Chains and Markov Decision Processes in Isabelle/HOL. *J. Autom. Reason.*, 2017. `doi:10.1007/s10817-016-9401-5`.

[30] Johannes Hölzl and Armin Heller. Three Chapters of Measure Theory in Isabelle/HOL. In *2nd International Conference on Interactive Theorem Proving (ITP)*, 2011. `doi:10.1007/978-3-642-22863-6_12`.

[31] Johannes Hölzl, Fabian Immler, and Brian Huffman. Type Classes and Filters for Mathematical Analysis in Isabelle/HOL. In *4th International Conference on Interactive Theorem Proving (ITP)*, 2013. `doi:10.1007/978-3-642-39634-2_21`.

[32] Richard Howey, Derek Long, and Maria Fox. VAL: Automatic Plan Validation, Continuous Effects and Mixed Initiative Planning Using PDDL. In *16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, 2004. `doi:10.1109/ICTAI.2004.120`.

[33] Fabian Immler. A Verified ODE Solver and the Lorenz Attractor. *J. Autom. Reason.*, 2018. `doi:10.1007/s10817-017-9448-y`.

[34] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In *29th International Conference Computer Aided Verification (CAV)*, 2017. `doi:10.1007/978-3-319-63387-9_5`.

[35] Alexander Krauss. *Automating Recursive Definitions and Termination Proofs in Higher-Order Logic*. PhD thesis, Technical University Munich, 2009. URL: `http://mediatum2.ub.tum.de/doc/681651/document.pdf`.

[36] Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. PDDL: The Planning Domain Definition Language, 1998.

[37] Drew V. McDermott. The 1998 AI Planning Systems Competition. *AI Mag.*, 2000. `doi:10.1609/aimag.v21i2.1506`.

[38] Robin Milner. Logic for computable functions description of a machine implementation. Technical report, Stanford University, 1972.

[39] Renato Neves, Luis S. Barbosa, Dirk Hofmann, and Manuel A. Martins. Continuity as a computational effect. *Journal of Logical and Algebraic Methods in Programming*, August 2016. `doi:10.1016/j.jlamp.2016.05.005`.

[40] Tobias Nipkow, Manuel Eberl, and Maximilian P. L. Haslbeck. Verified Textbook Algorithms - A Biased Survey. In *The 18th International Symposium on Automated Technology for Verification and Analysis (ATVA)*, 2020. `doi:10.1007/978-3-030-59152-6_2`.

[41] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*. 2002.

[42] Simon Roßkopf and Tobias Nipkow. A Formalization and Proof Checker for Isabelle's Metalogic. *J Autom Reasoning*, December 2022. `doi:10.1007/s10817-022-09648-w`.

[43] Scott Sanner. Relational Dynamic Influence Diagram Language (RDDL): Language Description.

[44] Maximilian Schäffeler and Mohammad Abdulaziz. Formally Verified Solution Methods for Infinite-Horizon Markov Decision Processes. In *The 37th AAAI Conference on Artificial Intelligence (AAAI)*, 2023. `doi:10.1609/aaai.v37i12.26759`.

[45] Maximilian Schäffeler and Mohammad Abdulaziz. Formally verified approximate policy iteration. In *Under Review*, 2024.

[46] Salomon Sickert and Jan Křetínský. MoChiBA: Probabilistic LTL Model Checking Using Limit-Deterministic Büchi Automata. In *Automated Technology for Verification and Analysis*, 2016. `doi:10.1007/978-3-319-46520-3_9`.

[47] Dmitriy Traytel, Andrei Popescu, and Jasmin Christian Blanchette. Foundational, Compositional (Co)datatypes for Higher-Order Logic: Category Theory Applied to Theorem Proving. In *The 27th Annual IEEE Symposium on Logic in Computer Science (LICS)*, 2012. `doi:10.1109/LICS.2012.75`.

[48] Niklaus Wirth. Program Development by Stepwise Refinement. *Commun. ACM*, 1971. `doi:10.1145/362575.362577`.

Requested Resources

In total, we ask for GBP267,808.60. The following is a breakdown of this figure.

```
Directly Incurred - Staff

  PDRA (6,36 @ 100\% FTE) GBP66,497.71

Directly Incurred - Non Staff

  Travel and Subsistence GBP3,000.00

Other Directly Incurred

  Conference Registration GBP2,000.00

  Laptop x 2  GBP7,300.00

  Visa and NHS Surcharge GBP1,751.00

Directly Allocated

  Staff

     Mohammad Abdulaziz Mansour GBP50,986.69

  Estate Costs GBP21,426.43

Indirect Costs GBP114,846.77

Total: GBP267,808.60
```

The following is the justification of the costs

Staff Costs:

PI: GBP50,986.69

The PI will spend 60% of his time on the project, for the project's entire duration, for which we ask for GBP50,986.69. This should include performing the tasks assigned to the PI, mainly the first work package, as well as guiding the PDRA in the second work package.

PDRA: GBP66,497.71

The project will also provide funding to the PDRA, who will be a 100% full time employee, who is expected to be a recent PhD graduate. Accordingly, the PDRA will be appointed at grade 6 spine point 36, which will cost GBP66,497.71 for the period of the project. The PDRA is expected to be employed for 1 year, and will be mainly spending his time, under the PI's guidance, on developing the verified validators and certificate checkers (WP 2 Task 1and WP 2 Task 2), in addition to collaborating with the PI on formalising the semantics.

Visa Costs (Total: GBP1,751.00):

We ask for the visa and Immigration health surcharge for the PDRA, who currently does not have permission to work in the UK.

Travel (Total: GBP5000):

Conferences are the major scientific events in which research in the areas of AI and verification is disseminated. Conferences which we plan to attend and submit our work to are the AAAI Conference on AI, which always takes place in North America, and the International Joint Conference on AI, the International Conference on Interactive Theorem Proving, and the International Conference on Automated Planning and Scheduling, all of whose locations change every year. Based on our experience attending these venues, we plan that, on average, every such trip will cost GBP2500, including the registration, travel and accommodation. Over the duration of the project the PI and the PDRA will travel to one such conference in 2025. This brings the conference travel costs to a total of GBP5000.

Computers (Total: GBP7248):

The computational needs of this project form the majority of the consumable costs. We need 1. computing servers on which we conduct computational experiments to evaluate the efficiency of the verified software, 2. powerful workstations for the development of formal proofs, which is a computationally demanding task, and 3. portable computers to present results in conferences and to colleagues during research visits. For the computing servers, we already have two servers procured as part of the PI's startup package provided by King's. For the development of formal proofs, we need machines, which have at least 64GB of memory, which is the currently recommended amount of memory for the Isabelle theorem prover. We found, based on our experience, that mobile workstations are perfectly suited to satisfy both, the development of formal proofs as well as presenting results in conferences and other similar events. One suitable such system is: Dell XPS 15 with a 13th Gen i9-13900HK CPU, 64GB of DDR5 RAM, and 1TB storage. This system costs around GBP3624, and since we plan to procure one such system for the PI and another for the PDRA, we request an overall cost of GBP7248.