R-PlanGPT: Neuro-Symbolic Plan Generation via Transformer-based Language Models

Massimiliano Tummolo[1,*], Mattia Chiari[1], Luca Putelli[1], Nicholas Rossetti[1], Ivan Serina[1] and Alfonso Emilio Gerevini[1]

[1]University of Brescia, Brescia, Italy, via Branze 38

Abstract

R-PlanGPT is a neuro-symbolic architecture designed to generate solution plans for classical planning problems by learning from examples. It combines a generative model (PlanGPT), a symbolic validator (VAL) and a classical planner (LPG). PlanGPT learns to solve new instances within the same domain as a general policy, treating planning as a generative task and producing action sequences given the initial state and the goal of a problem. In order to guarantee the correctness of the neural model output, VAL is called to validate every plan produced by PlanGPT. If the solution is not valid, it is repaired by LPG. We demonstrate the capabilities of R-PlanGPT on standard planning benchmarks, highlighting its ability to generate valid, high-quality plans.

1. Introduction

Recent advancements in Large Language Models (LLMs) have shown remarkable performance across various natural language processing tasks [1, 2]. However, a broader use of these technologies includes mathematical inference tasks [3], and code writing [4]. In terms of reasoning abilities, although there is a basic understanding that these model are capable or common-sense reasoning [5], an important benchmark for these abilities is automated planning [6] and, in particular, solving planning problems [7, 8, 9, 10].

In this demo, we present R-PlanGPT, a neuro-symbolic architecture that addresses plan generation as a sequence modeling task using a GPT-based architecture. As it can be seen in Figure 1, the system is composed by three main modules: PlanGPT, a GPT-model trained from scratch on classical planning problems (expressed in PDDL [11]) which learns to generate action sequences that solve planning instances in a given domain. Since there is no theoretical guarantee that the neural model provides the correct solution of the problem, R-PlanGPT includes a validator [12, 13] to ensure plan soundness. If the solution is not valid, R-PlanGPT invokes a classical planner [14] to repair the output of the neural model and to provide a valid plan [15]. We test R-PlanGPT on several benchmark domains from the International Planning Competition (IPC) [16]. Our results show that although the GPT-model reaches good results by itself, the inclusion of the symbolic components further increase the performance, making R-PlanGPT capable of generating valid high-quality plans.

2. Background

Automated Planning is a branch of Artificial Intelligence focused on generating a sequence of actions (a plan) that an agent can perform to transition from an initial state to a goal state, given a formal model of the domain [17]. While classical planners focus on solving individual problem instances, Generalized Planning (GP) instead seeks to derive general policies that solve several problems within a domain.

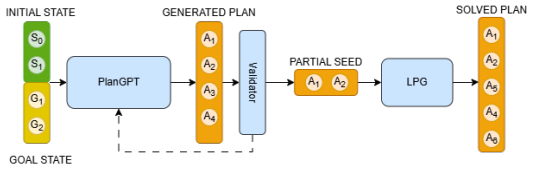

Figure 1: Example of input/output for a planning problem with two fluents in the initial state (F_0 and F_1) and two fluents forming the goal (G_0 and G_1). PlanGPT generates a sequence of actions (A_1, A_2, A_3, A_4), which is passed to a symbolic validator. If the plan is invalid, a valid prefix (e.g., A_1, A_2) is extracted, and the LPG planner is used to repair or complete the plan. The final plan returned is valid and complete: (A_1, A_2, A_5, A_4, A_6).

A general policy [18] is a mapping from states (or observations) to actions, enabling an agent to solve previously unseen problems without having to compute each solution from scratch. For instance, in the well-known blocksworld domain, a general policy might instruct the agent to "clear all blocks and then stack them in goal order," regardless of the number of blocks involved. Recent learning-based approaches have explored extracting general policies from solved examples, often using neural networks such as CNNs or GNNs [19, 20, 21]. However, these typically require domain-specific encoding and provide limited expressivity or scalability. On the other hand, Transformer-based language models have demonstrated strong capabilities in sequence modelling tasks and show potential for learning policies from data without handcrafted features [8, 22, 10].

3. Methodology

This section details the pipeline for training and using R-PlanGPT for classical planning tasks. As shown in Figure 1, the system is composed of a transformer-based model (PlanGPT [23]) a symbolic validator (VAL [12]), and the LPG planner [14].

First, we create a dataset composed of solved planning instances from classical domains written in PDDL to train the neural component; these are generated using standard domain-specific generators, following the IPC conventions to ensure a range of complexity. Each problem is solved using the LPG planner, and we collect up to four plans per instance to provide diversity. To avoid overfitting to naming conventions, object names in problems and plans are randomized.

ANSyA 2025: 1st International Workshop on Advanced Neuro-Symbolic Applications, co-located with ECAI 2025.

*Corresponding Author. Email: massimiliano.tummolo@unibs.it. M. Tummolo was enrolled in the National Doctorate on AI conducted by Sapienza, University of Rome with the University of Brescia.

Next, we train from scratch the neural component of the system, PlanGPT, which is based on a GPT architecture and receives as input a textual prompt encoding the initial state and goal of the planning problem. The model is trained with standard cross-entropy loss to predict the next token in a plan sequence. To prevent overfitting, we use a custom early stopping criterion called Coverage Early Stopping [23], which terminates training when the percentage of valid plans generated on the validation set stabilizes.

At inference time, after the training, PlanGPT autoregressively generates grounded action sequences. However, PlanGPT can generate actions with unmet preconditions or fails to complete all the goals, thus generating an invalid plan. To check if a plan produced by PlanGPT is a valid solution, we incorporate a symbolic validator (VAL) to assess the correctness of each generated action and to verify that each goal is satisfied at the end of the generation. To prevent the model from generating non-applicable actions, we further introduce the Validated Multi-Beam Search (VAL-MB) strategy, which integrates VAL into the decoding process by validating candidate actions during generation and pruning invalid beams on the fly. Finally, if the generated output is invalid or incomplete, we apply a plan repair strategy using LPG. In this setup, a valid plan prefix is extracted and provided as a seed to LPG, which continues the search and produces a complete solution, combining the strengths of neural generation with symbolic reasoning.

4. System Demonstration

As shown in Figure 2, the demo presents an interactive system designed to showcase how to generate valid plans using R-PlanGPT. The platform allows users to interact with the whole pipeline, from problem specification to plan generation and validation.

The user begins by selecting a classical planning domain from a predefined set (e.g., blocksworld, logistics) and uploads a corresponding PDDL problem file, which includes the initial state and goal. The system automatically parses and verifies the syntactic correctness of the uploaded file against the domain definition to ensure compatibility. This procedure includes two key steps: (i) a conversion mechanism that remaps object names not present in the model's vocabulary to randomly selected placeholder names from the vocabulary, ensuring compatibility with the model's tokenization; and (ii) a check on the number of objects present in the problem. Since PlanGPT is trained with a fixed vocabulary size and limited object capacity, the system ensures that the number of objects in the problem does not exceed the maximum supported.If either check fails, the user receives an error message and is asked to revise the input.

Once validated, the user can trigger the plan generation step using PlanGPT. The system supports various generation strategies: greedy decoding, multi-beam search, sampling, or the Validated Multi-Beam approach. In VAL-MB, the symbolic validator is invoked at each generation step to discard invalid actions on the fly, enforcing plan soundness during decoding. For other decoding strategies, validation is performed after the complete plan has been generated. If the generated plan is deemed valid by the VAL tool, it is directly displayed to the user as a viable solution.

Otherwise, the validator identifies the first precondition violation, and the system extracts the longest valid plan prefix. At this point, the user is offered the option to invoke

Figure 2: Screenshot of the demo page.

Domain	FD	LAMA	LPG	PlanGPT	R-PlanGPT
blocksworld	27.61	33.93	**35.00**	34.73	34.73
depots	15.15	19.45	18.59	17.42	**21.40**
driverlog	18.79	**19.73**	18.59	17.08	17.46
floortile	2.00	2.00	17.36	**19.64**	**19.64**
logistics	26.41	26.55	24.99	14.02	**27.12**
satellite	18.51	18.64	**19.74**	11.94	19.68
visitall	19.07	**19.60**	19.46	16.53	17.46
zenotravel	18.85	**19.82**	19.22	15.71	17.51
total	146.39	159.72	172.95	147.07	**175.00**

Table 1
Results in terms of IPCQ of R-PlanGPT, with respect to classical planners (FD, LAMA and LPG) and the neural model PlanGPT.

LPGto repair or complete the plan. LPG uses the prefix as a seed to guide its search process, producing a valid plan more efficiently than starting from scratch.

Once the solution is obtained, the output is displayed, and the user is shown the generated plan, a summary of the selected domain and problem, and the validation results if requested.

5. System Evaluation

We evaluate R-PlanGPT on a suite of classical planning benchmarks, demonstrating its effectiveness. The IPCScore-Quality (IPCQ) metric evaluates the quality of solutions of a planning system by comparing each plan's cost to the best-known plan for the same problem. Higher scores indicate plans that are closer to optimal, while unsolved problems score zero. As shown in Table 1, R-PlanGPT outperforms all other approaches in terms of IPCQ, achieving the highest overall score of 175.00 on the IPC domains. This result demonstrates that combining PlanGPT with LPG as a post-repair step leads to plans of higher quality than either PlanGPT or symbolic planners alone. Despite the good performance of the neural model by itself, the integration of a symbolic planner improves it in domains where PlanGPT alone struggled, such as logistics, satellite, and depots. Moreover, it reaches the performance of LPG and LAMA in most domains.

6. Conclusion

In this demo, we have presented R-PlanGPT, a system that integrates language models with symbolic tools for solving classical planning problems. Future developments will focus on supporting nondeterministic planning, incorporating

macro-actions and temporal logic, enhancing interpretability, or other forms of neuro-symbolic integration [24, 25, 26].

7. Acknowledgements

This work has been supported by: MUR (Italian Ministry of University and Research) PRIN-2020 project RIPER (n. 20203FFYLK); PNRR MUR project PE0000013-FAIR, cascade funding call, ResilientPlans; AI4WATER project, part of the PRIMA Programme supported by the European Union and by MUR; and by and by Regione Lombardia through the initiative "Programma degli interventi per la ripresa economica: sviluppo di nuovi accordi di collaborazione con le università per la ricerca, l'innovazione e il trasferimento tecnologico" - DGR n. XI/4445/2021.

References

[1] A. Radford, K. Narasimhan, Improving language understanding by generative pre-training, in: preprint, 2018. api.semanticscholar.org/CorpusID:49313245.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: NIPS, 2017, pp. 5998–6008.

[3] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, D. Zhou, Chain-of-thought prompting elicits reasoning in large language models, in: NeurIPS, 2022.

[4] Y. Wang, W. Wang, S. R. Joty, S. C. H. Hoi, Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation, in: EMNLP (1), Association for Computational Linguistics, 2021, pp. 8696–8708.

[5] M. Geva, D. Khashabi, E. Segal, T. Khot, D. Roth, J. Berant, Did Aristotle use a laptop? A question answering benchmark with implicit reasoning strategies, Trans. Assoc. Comput. Linguistics 9 (2021) 346–361.

[6] V. Pallagani, B. C. Muppasani, K. Roy, F. Fabiano, A. Loreggia, K. Murugesan, B. Srivastava, F. Rossi, L. Horesh, A. P. Sheth, On the prospects of incorporating large language models (llms) in automated planning and scheduling (APS), in: ICAPS, AAAI Press, 2024, pp. 432–444.

[7] M. Chiari, L. Putelli, N. Rossetti, I. Serina, A. E. Gerevini, On planning through llms, in: ICAPS, AAAI Press, 2025.

[8] V. Pallagani, B. Muppasani, B. Srivastava, F. Rossi, L. Horesh, K. Murugesan, A. Loreggia, F. Fabiano, R. Joseph, Y. Kethepalli, Plansformer tool: Demonstrating generation of symbolic plans using transformers, in: IJCAI, IJCAI Org., 2023, pp. 7158–7162.

[9] L. Serina, M. Chiari, A. E. Gerevini, L. Putelli, I. Serina, A preliminary study on BERT applied to automated planning, in: IPS/RiCeRcA/SPIRIT@AI*IA, volume 3345 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022.

[10] K. Valmeekam, M. Marquez, A. O. Hernandez, S. Sreedharan, S. Kambhampati, Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change, in: NeurIPS, 2023.

[11] D. McDermott, M. Ghallab, A. E. Howe, C. A. Knoblock, A. Ram, M. M. Veloso, D. S. Weld, D. E. Wilkins, Pddl-the planning domain definition language, 1998. URL: https://api.semanticscholar.org/CorpusID:59656859.

[12] R. Howey, D. Long, M. Fox, VAL: automatic plan validation, continuous effects and mixed initiative planning using PDDL, in: ICTAI, IEEE Computer Society, 2004, pp. 294–301.

[13] N. Rossetti, M. Tummolo, A. E. Gerevini, L. Putelli, I. Serina, M. Olivato, Enhancing gpt-based planning policies by model-based plan validation, Proceedings of the 18th International Conference on Neural-Symbolic Learning and Reasoning (2024).

[14] A. Gerevini, I. Serina, LPG: A planner based on local search for planning graphs with action costs, in: AIPS, AAAI Press, 2002, pp. 13–22.

[15] M. Tummolo, N. Rossetti, A. E. Gerevini, L. Putelli, I. Serina, M. Olivato, Integrating classical planners with gpt-based planning policies, in: AI*IA, Lecture Notes in Computer Science, Springer, 2024.

[16] A. Taitler, R. Alford, J. Espasa, G. Behnke, D. Fiser, M. Gimelfarb, F. Pommerening, S. Sanner, E. Scala, D. Schreiber, J. Segovia-Aguas, J. Seipp, The 2023 international planning competition, AI Mag. 45 (2024) 280–296.

[17] M. Ghallab, D. S. Nau, P. Traverso, Automated planning - theory and practice, Elsevier, 2004.

[18] Y. Hu, G. De Giacomo, Generalized planning: Synthesizing plans that work for multiple environments, in: IJCAI, IJCAI Org., 2011, pp. 918–923.

[19] E. Groshev, M. Goldstein, A. Tamar, S. Srivastava, P. Abbeel, Learning generalized reactive policies using deep neural networks, in: ICAPS, AAAI Press, 2018, pp. 408–416.

[20] S. Ståhlberg, B. Bonet, H. Geffner, Learning general optimal policies with graph neural networks: Expressive power, transparency, and limits, in: ICAPS, AAAI Press, 2022, pp. 629–637.

[21] S. Toyer, S. Thiébaux, F. W. Trevizan, L. Xie, Asnets: Deep learning for generalised planning, J. Artif. Intell. Res. 68 (2020) 1–68.

[22] T. Silver, S. Dan, K. Srinivas, J. B. Tenenbaum, L. P. Kaelbling, M. Katz, Generalized planning in PDDL domains with pretrained large language models, in: AAAI, AAAI Press, 2024, pp. 20256–20264.

[23] N. Rossetti, M. Tummolo, A. E. Gerevini, L. Putelli, I. Serina, M. Chiari, M. Olivato, Learning general policies for planning through gpt models, Proceedings of the International Conference on Automated Planning and Scheduling 34 (2024) 500–508.

[24] M. Chiari, A. E. Gerevini, F. Percassi, L. Putelli, I. Serina, M. Olivato, Goal recognition as a deep learning task: The grnet approach, in: ICAPS, AAAI Press, 2023, pp. 560–568.

[25] M. Chiari, A. E. Gerevini, A. Loreggia, L. Putelli, I. Serina, Fast and slow goal recognition, in: M. Dastani, J. S. Sichman, N. Alechina, V. Dignum (Eds.), Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2024, Auckland, New Zealand, May 6-10, 2024, International Foundation for Autonomous Agents and Multiagent Systems / ACM, 2024, pp. 354–362.

[26] L. Serina, M. Chiari, A. E. Gerevini, L. Putelli, I. Serina, Towards efficient online goal recognition through deep learning, in: AAMAS, International Foundation for Autonomous Agents and Multiagent Systems / ACM, 2025, pp. 1895–1903.