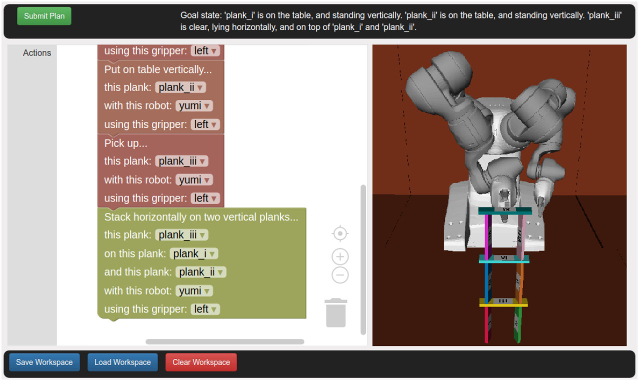
JEDAI: A System for Skill-Aligned Explainable Robot Planning[*]

Naman Shah[†], **Pulkit Verma**[†], **Trevor Angle,** and **Siddharth Srivastava**

Autonomous Agents and Intelligent Robots Lab,
School of Computing and Augmented Intelligence, Arizona State University, AZ, USA
{shah.naman, verma.pulkit, taangle, siddharths}@asu.edu

Abstract

This paper presents JEDAI, an AI system designed for outreach and educational efforts aimed at non-AI experts. JEDAI features a novel synthesis of research ideas from integrated task and motion planning and explainable AI. JEDAI helps users create high-level, intuitive plans while ensuring that they will be executable by the robot. It also provides users customized explanations about errors and helps improve their understanding of AI planning as well as the limits and capabilities of the underlying robot system.



Goal state: 'plank_i' is on the table, and standing vertically. 'plank_ii' is on the table, and standing vertically. 'plank_iii' is clear, lying horizontally, and on top of 'plank_i' and 'plank_ii'.

using this gripper: left
Put on table vertically...
this plank: plank_ii
with this robot: yumi
using this gripper: left
Pick up...
this plank: plank_iii
with this robot: yumi
using this gripper: left
Stack horizontally on two vertical planks...
this plank: plank_iii
on this plank: plank_i
and this plank: plank_ii
with this robot: yumi
using this gripper: left

Figure 1: JEDAI system with a Blockly-based plan creator on the left and a simulator window on the right.

1 Introduction

AI systems are increasingly common in everyday life, where they can be used by laypersons who may not understand how these autonomous systems work or what they can and cannot do. This problem is particularly salient in cases of taskable AI systems whose functionality can change based on the tasks they are performing. Lack of understanding about the limits of an imperfect system can result in unproductive usage or, in the worst-case, serious accidents (Randazzo 2018). This, in turn, limits the adoption and productivity of the AI systems. The current research on AI safety focuses on designing AI systems that allow humans to safely instruct and control them (e.g., (Russell, Dewey, and Tegmark 2015; Zilberstein 2015; Hadfield-Menell et al. 2016, 2017; Russell 2017)). In this work, we present an AI system JEDAI (JEDAI Explains Decision-Making AI) that can be used in outreach and educational efforts to help laypersons learn how to provide AI systems with new tasks, debug such systems, and understand their capabilities.

The research ideas brought together in JEDAI address three key technical challenges: (i) abstracting a robot's functionalities into high-level actions (capabilities) that the user can more easily understand; (ii) converting the user-understandable capabilities into low-level motion plans that a robot can execute; and (iii) explaining errors in a manner sensitive to the user's current level of knowledge so as to make the robot's capabilities and limitations clear.

JEDAI utilizes recent work in explainable AI and integrated task and motion planning to address these challenges and provides a simple interface to support accessibility. Users select a domain and an associated task, after which they create a plan consisting of high-level actions (Fig. 1 left) to complete the task. The user puts together a plan in a drag-and-drop workspace, built with the Blockly visual programming library (Google 2017). JEDAI validates this plan using the Hierarchical Expertise Level Modeling algorithm (HELM) (Sreedharan, Srivastava, and Kambhampati 2018, 2021). If the plan contains any errors, HELM computes a user-specific explanation of why the plan would fail. JEDAI converts such explanations to natural language, thus helping to identify and fix any gaps in the user's understanding. Whereas, if the plan given by the user is a correct solution to the current task, JEDAI uses a task and motion planner ATM-MDP (Shah et al. 2020; Shah and Srivastava 2021) to convert the high-level plan, that the user understands, to a low-level motion plan that the robot can execute. The user is shown the execution of this low-level motion plan by the robot in a simulated environment (Fig. 1 right).

The next section discusses the relationship of the presented methods with prior work. Sec. 3 presents an architecture of JEDAI. Finally, Sec. 4 presents our conclusion and future directions.

[*] Also appearing in AAMAS 2022 Demonstration Track as Shah et al. (2022)

[†] Equal contribution. Alphabetical order.

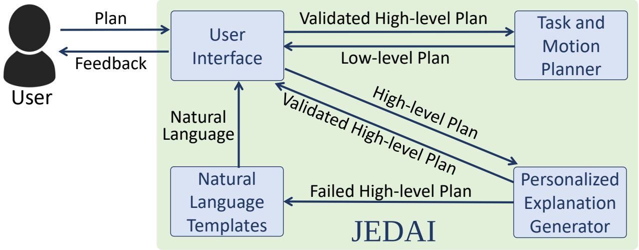


Figure 2: Architecture of JEDAI showing interaction between the four core components.

2 Related Work

Prior work on the topic includes approaches that solve the three technical challenges mentioned earlier *in isolation*. This includes tools for: providing visualizations or animations of standard planning domains (Magnaguagno et al. 2017; Chen et al. 2019; Aguinaldo and Regli 2021; Dvorak, Agarwal, and Baklanov 2021; De Pellegrin and Petrick 2021; Roberts et al. 2021); making it easier for non-expert users to program robots with low-level actions (Krishnamoorthy and Kapila 2016; Weintrop et al. 2018; Huang et al. 2020; Winterer et al. 2020); and generating explanations for plans provided by the users (Grover et al. 2020; Karthik et al. 2022; Brandao et al. 2021; Kumar et al. 2022). In addition, none of these works make the instructions easier for the user, have the ability to automatically compute user-aligned explanations, and work with real robots (or their simulators) at the same time. JEDAI addresses all three challenges in tandem by using 3D simulations for domains with real robots and their actual constraints and providing personalized explanations that inform a user of any mistake they make while using the system.

3 Architecture

Fig. 2 shows the four core components of the JEDAI framework: (i) user interface, (ii) task and motion planner, (iii) personalized explanation generator, and (iv) natural language templates. We now describe each component in detail.

3.1 User interface

JEDAI's user interface (Fig. 1) is made to be unintimidating and easy to use. The Blockly visual programming interface is used to facilitate this. JEDAI generates a separate interconnecting block for each high-level action, and action parameters are picked from drop-down selection fields that display type-consistent options for each parameter. Users can drag-and-drop these actions and select different arguments to create a high-level plan.

3.2 Personalized explanation generator

Users will sometimes make mistakes when planning, either failing to achieve goal conditions or applying actions before the necessary preconditions are satisfied. For inexperienced users in particular, these mistakes may stem from an incomplete understanding of the task's requirements or the robot's capabilities. JEDAI assists users in apprehending these details by providing explanations personalized to each user.

Explanations in the context of this work are of two types: (i) non-achieved goal conditions, and (ii) violation of a precondition of an action. JEDAI validates the plan submitted by the user to check if it achieves all goal conditions. If it fails to achieve any goal condition, the user is informed about it. E.g., consider the *pick-up* action shown in Fig. 1 fails if the robot is not holding the *plank_ii*, this error is explained to the user as "The action at step 3 (pick up plank 'plank_ii' with robot 'yumi' using gripper 'left') could not be performed because 'plank_ii' is not held in 'left' gripper."

JEDAI uses HELM to compute such user-specific contrastive explanations in order to explain any unmet precondition in an action used in the user's plan. HELM does this by using the plan submitted by the user to estimate the user's understanding of the robot's model and then uses the estimated model to compute the personalized explanations. In case of multiple errors in the user's plan, HELM generates explanation for one of the errors. This is because explaining the reason for more than one errors might be unnecessary and in the worst case might leave the user feeling overwhelmed (Miller 2019). An error is selected for explanation by HELM based on optimizing a cost function that indicates the relative difficulty of concept understandability which can be changed to reflect different users' background knowledge.

3.3 Natural language templates

Even with a user-friendly interface and personalized explanations for errors in abstract plans, domain model syntax used for interaction with ATM-MDP presents a significant barrier to a non-expert trying to understand the state of an environment and the capabilities of a robot. To alleviate this, JEDAI uses language templates that use the structure of the planning formalism for generating natural language descriptions for goals, actions, and explanations. E.g., the action *"pickup (plank_i yumi gripper_left)"* can be described in natural language as "pick up *plank_i* using robot *yumi* with *the left gripper*". Currently, we use hand-written templates for these translations, but an automated approach can also be used.

3.4 Task and motion planner

JEDAI uses ATM-MDP to convert the high-level plan submitted by the user into sequences of low-level primitive actions that a robot can execute.

ATM-MDP uses sampling-based motion planners to provide a probabilistically complete approach to hierarchical planning. High-level plans are refined by computing feasible motion plans for each high-level action. If an action does not accept any valid refinement due to discrepancies between the symbolic state and the low-level environment, it reports the failure back to JEDAI. If all actions in the high-level plan are refined successfully, the plan's execution is shown using the OpenRAVE simulator (Diankov and Kuffner 2008).

3.5 Implementation

Any custom domain can be set up with JEDAI. We provide five built-in domains, each with one of YuMi (ABB 2015) or Fetch (Wise et al. 2016) robots. Each domain contains a set of problems that the users can attempt to solve and low-level environments corresponding to these problems. Source code for the framework, an already setup virtual machine, and the documentation are available at: https://github.com/aair-lab/AAIR-JEDAI. A video demonstrating JEDAI's working is available at: https://youtu.be/ASIg28-ADZ8.

4 Conclusions and Future Work

We demonstrated a novel AI tool JEDAI for helping people understand the capabilities of an arbitrary AI system and enabling them to work with such systems. JEDAI converts the user's input plans to low level motion plans executable by the robot if it is correct, or explains to the user any error in the plan if it is incorrect. JEDAI works with off-the-shelf task and motion planners and explanation generators. This structure allows it to scale automatically with improvements in either of these active research areas. JEDAI's vizualization-based interface could also be used to foster trust in AI systems (Beauxis-Aussalet et al. 2021).

JEDAI uses predefined abstractions to verify plans provided by the user. In the future, we plan on extending it to learn abstractions automatically (Shah and Srivastava 2022). JEDAI could also be extended as an interface for assessing an agent's functionalities and capabilities by interrogating the agent (Verma, Marpally, and Srivastava 2021; Nayyar, Verma, and Srivastava 2022; Verma, Marpally, and Srivastava 2022) as well as to work as an interface that makes AI systems compliant with Level II assistive AI – systems that makes it easy for operators to learn how to use them safely (Srivastava 2021). Extending this tool for working in non-stationary settings, and generating natural language descriptions of predicates and actions autonomously are a few other promising directions of future work.

Acknowledgements

We thank Kiran Prasad and Kyle Atkinson for help with the implementation, Sarath Sreedharan for help with setting up HELM, and Sydney Wallace for feedback on user interface design. We also thank Chirav Dave, Rushang Karia, Judith Rosenke, and Amruta Tapadiya for their work on an earlier version of the system. This work was supported in part by the NSF grants IIS 1909370, IIS 1942856, IIS 1844325, OIA 1936997, and the ONR grant N00014-21-1-2045.

References

ABB. 2015. ABB YuMi - IRB 14000. https://new.abb.com/products/robotics/collaborative-robots/irb-14000-yumi.

Aguinaldo, A.; and Regli, W. 2021. A Graphical Model-Based Representation for Classical AI Plans using Category Theory. In *ICAPS 2021 Workshop on Explainable AI Planning*.

Beauxis-Aussalet, E.; Behrisch, M.; Borgo, R.; Chau, D. H.; Collins, C.; Ebert, D.; El-Assady, M.; Endert, A.; Keim, D. A.; Kohlhammer, J.; Oelke, D.; Peltonen, J.; Riveiro, M.; Schreck, T.; Strobelt, H.; and van Wijk, J. J. 2021. The Role of Interactive Visualization in Fostering Trust in AI. *IEEE Computer Graphics and Applications*, 41(6): 7–12.

Brandao, M.; Canal, G.; Krivić, S.; and Magazzeni, D. 2021. Towards Providing Explanations for Robot Motion Planning. In *Proc. ICRA*.

Chen, G.; Ding, Y.; Edwards, H.; Chau, C. H.; Hou, S.; Johnson, G.; Sharukh Syed, M.; Tang, H.; Wu, Y.; Yan, Y.; Gil, T.; and Nir, L. 2019. Planimation. In *ICAPS 2019 System Demonstrations*.

De Pellegrin, E.; and Petrick, R. P. A. 2021. PDSim: Simulating Classical Planning Domains with the Unity Game Engine. In *ICAPS 2021 System Demonstrations*.

Diankov, R.; and Kuffner, J. 2008. OpenRAVE: A Planning Architecture for Autonomous Robotics. Technical Report CMU-RI-TR-08-34, Carnegie Mellon University, Pittsburgh, PA, USA.

Dvorak, F.; Agarwal, A.; and Baklanov, N. 2021. Visual Planning Domain Design for PDDL using Blockly. In *ICAPS 2021 System Demonstrations*.

Google. 2017. Blockly. https://github.com/google/blockly.

Grover, S.; Sengupta, S.; Chakraborti, T.; Mishra, A. P.; and Kambhampati, S. 2020. RADAR: Automated Task Planning for Proactive Decision Support. *Human–Computer Interaction*, 35(5-6): 387–412.

Hadfield-Menell, D.; Dragan, A.; Abbeel, P.; and Russell, S. 2017. The Off-Switch Game. In *Proc. IJCAI*.

Hadfield-Menell, D.; Russell, S. J.; Abbeel, P.; and Dragan, A. 2016. Cooperative Inverse Reinforcement Learning. In *Proc. NeurIPS*.

Huang, G.; Rao, P. S.; Wu, M.-H.; Qian, X.; Nof, S. Y.; Ramani, K.; and Quinn, A. J. 2020. Vipo: Spatial-Visual Programming with Functions for Robot-IoT Workflows. In *Proc. CHI*.

Karthik, V.; Sreedharan, S.; Sengupta, S.; and Kambhampati, S. 2022. RADAR-X: An Interactive Mixed Initiative Planning Interface Pairing Contrastive Explanations and Revised Plan Suggestions. In *Proc. ICAPS*.

Krishnamoorthy, S. P.; and Kapila, V. 2016. Using A Visual Programming Environment and Custom Robots to Learn C Programming and K-12 STEM Concepts. In *Proceedings of the 6th Annual Conference on Creativity and Fabrication in Education*.

Kumar, A.; Vasileiou, S. L.; Bancilhon, M.; Ottley, A.; and Yeoh, W. 2022. VizXP: A Visualization Framework for Conveying Explanations to Users in Model Reconciliation Problems. In *Proc. ICAPS*.

Magnaguagno, M. C.; Fraga Pereira, R.; Móre, M. D.; and Meneguzzi, F. R. 2017. WEB PLANNER: A Tool to Develop Classical Planning Domains and Visualize Heuristic State-Space Search. In *ICAPS 2017 Workshop on User Interfaces and Scheduling and Planning*.

Miller, T. 2019. Explanation in Artificial Intelligence: Insights from the Social Sciences. *Artificial Intelligence*, 267: 1–38.

Nayyar, R. K.; Verma, P.; and Srivastava, S. 2022. Differential Assessment of Black-Box AI Agents. In *Proc. AAAI*.

Randazzo, R. 2018. What went wrong with Uber's Volvo in fatal crash? Experts shocked by technology failure. *The AZ Republic*.

Roberts, J. O.; Mastorakis, G.; Lazaruk, B.; Franco, S.; Stokes, A. A.; and Bernardini, S. 2021. vPlanSim: An Open Source Graphical Interface for the Visualisation and Simulation of AI Systems. In *Proc. ICAPS*.

Russell, S. 2017. Provably Beneficial Artificial Intelligence. *The Next Step: Exponential Life*.

Russell, S.; Dewey, D.; and Tegmark, M. 2015. Research Priorities for Robust and Beneficial Artificial Intelligence. *AI Magazine*, 36(4): 105–114.

Shah, N.; Kala Vasudevan, D.; Kumar, K.; Kamojjhala, P.; and Srivastava, S. 2020. Anytime Integrated Task and Motion Policies for Stochastic Environments. In *Proc. ICRA*.

Shah, N.; and Srivastava, S. 2021. Anytime Stochastic Task and Motion Policies. *arXiv preprint arXiv:2108.12537*.

Shah, N.; and Srivastava, S. 2022. Using Deep Learning to Bootstrap Abstractions for Hierarchical Robot Planning. In *Proc. AAMAS*.

Shah, N.; Verma, P.; Angle, T.; and Srivastava, S. 2022. JEDAI: A System for Skill-Aligned Explainable Robot Planning. In *Proc. AAMAS*.

Sreedharan, S.; Srivastava, S.; and Kambhampati, S. 2018. Hierarchical Expertise Level Modeling for User-Specific Contrastive Explanations. In *Proc. IJCAI*.

Sreedharan, S.; Srivastava, S.; and Kambhampati, S. 2021. Using State Abstractions to Compute Personalized Contrastive Explanations for AI Agent Behavior. *Artificial Intelligence*, 301: 103570.

Srivastava, S. 2021. Unifying Principles and Metrics for Safe and Assistive AI. In *Proc. AAAI*.

Verma, P.; Marpally, S. R.; and Srivastava, S. 2021. Asking the Right Questions: Learning Interpretable Action Models Through Query Answering. In *Proc. AAAI*.

Verma, P.; Marpally, S. R.; and Srivastava, S. 2022. Discovering User-Interpretable Capabilities of Black-Box Planning Agents. In *AAAI 2022 Workshop on Explainable Agency in Artificial Intelligence*.

Weintrop, D.; Afzal, A.; Salac, J.; Francis, P.; Li, B.; Shepherd, D. C.; and Franklin, D. 2018. Evaluating CoBlox: A Comparative Study of Robotics Programming Environments for Adult Novices. In *Proc. CHI*.

Winterer, M.; Salomon, C.; Köberle, J.; Ramler, R.; and Schittengruber, M. 2020. An Expert Review on the Applicability of Blockly for Industrial Robot Programming. In *Proceedings of the 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*.

Wise, M.; Ferguson, M.; King, D.; Diehr, E.; and Dymesich, D. 2016. Fetch and Freight: Standard Platforms for Service Robot Applications. In *IJCAI 2016 Workshop on Autonomous Mobile Service Robots*.

Zilberstein, S. 2015. Building Strong Semi-Autonomous Systems. In *Proc. AAAI*.