Poster: How to Raise a Robot — Beyond Access Control Constraints in Assistive Humanoid Robots

Niklas Hemken
niklas.hemken@student.kit.edu
KIT
Karlsruhe, Germany

Florian Jacob
florian.jacob@kit.edu
KIT
Karlsruhe, Germany

Fabian Peller-Konrad
fabian.peller-konrad@kit.edu
KIT
Karlsruhe, Germany

Rainer Kartmann
rainer.kartmann@kit.edu
KIT
Karlsruhe, Germany

Tamim Asfour
asfour@kit.edu
KIT
Karlsruhe, Germany

Hannes Hartenstein
hannes.hartenstein@kit.edu
KIT
Karlsruhe, Germany

ABSTRACT

Humanoid robots will be able to assist humans in their daily life, in particular due to their versatile action capabilities. However, while these robots need a certain degree of autonomy to learn and explore, they also should respect various constraints, for access control and beyond. We explore incorporating privacy and security constraints (Activity-Centric Access Control and Deep Learning Based Access Control) with robot task planning approaches (classical symbolic planning and end-to-end learning-based planning). We report preliminary results on their respective trade-offs and conclude that a hybrid approach will most likely be the method of choice.

CCS CONCEPTS

• **Security and privacy** → **Access control**; • **Computer systems organization** → **Robotics**; • **Computing methodologies** → **Robotic planning**.

KEYWORDS

Humanoid Robots, Robot Task Planning, Activity-Centric Access Control, Deep Learning based Access Control

ACM Reference Format:
Niklas Hemken, Florian Jacob, Fabian Peller-Konrad, Rainer Kartmann, Tamim Asfour, and Hannes Hartenstein. 2023. Poster: How to Raise a Robot — Beyond Access Control Constraints in Assistive Humanoid Robots. In *Proceedings of the 28th ACM Symposium on Access Control Models and Technologies (SACMAT '23), June 7–9, 2023, Trento, Italy*. ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3589608.3595078

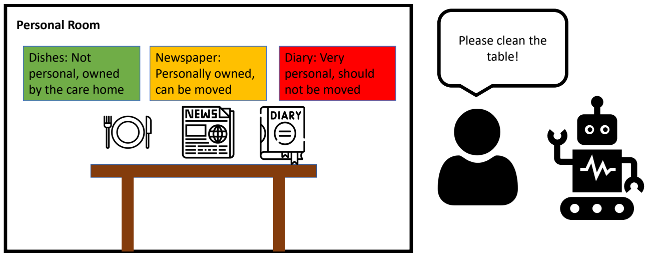
Personal Room
Dishes: Not personal, owned by the care home
Newspaper: Personally owned, can be moved
Diary: Very personal, should not be moved
NEWS
DIARY
Please clean the table!

Figure 1: An assistive robot is tasked to clean the table in the personal room of a care home resident. The challenge is to integrate an abstract "do not move private objects" policy as specific constraints into robot task planning.

1 INTRODUCTION AND RELATED WORK

The potential of humanoid robots to assist humans lies in the ability to learn whatever is needed for assistance. However, 'whatever is needed' has to be restricted to safeguard safety, security, and privacy policies and preferences. The challenge for classical approaches, i.e., based on symbols and logical formulas, to ensure constraints lies in the task universality of humanoid robots. Their wide range of tasks and deployment favors approaches in which the robot is granted increased sovereignty and should learn its authorizations in the field. Learning promises scalability of problem complexity up to task universality, while keeping manual specification complexity manageable for humans. However, learning changes the nature of access control from certain enforcement to probable observance.

In a first attempt to tackle this problem of balancing sovereignty and 'obedience', we proposed a joint modeling framework for robot task planning and access control [1]. By directly incorporating access control into task planning, robots are unable to even "think about" forbidden behavior. While the fit between the notion of an activity in Activity-Centric Access Control (ACAC) [2] and task planning was shown in theory, we now report on preliminary practical insights. Recently, machine-learning-based approaches to assist or even completely perform access decisions have been proposed, as seen with Machine-Learning- and Deep-Learning-based Access Control (MLBAC / DLBAC) [4]. In this poster abstract, we investigate how the logical and neural approaches to robot task planning can ensure logical and neural constraints beyond access control, discuss their trade-offs, and propose a hybrid approach.

As toy example illustrated in Fig. 1, a robot is tasked to clean a table in a personal room, for which the robot now needs a plan. It considers whether it is allowed to move objects on the table, i.e., the authorization to move an object is a precondition of the action for moving that object. We assume constraints that formalize policies like "do not move personal items," which allow, e.g., the removal of care home owned dishes, but disallow moving a diary.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SACMAT '23, June 7–9, 2023, Trento, Italy
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0173-3/23/06.
https://doi.org/10.1145/3589608.3595078

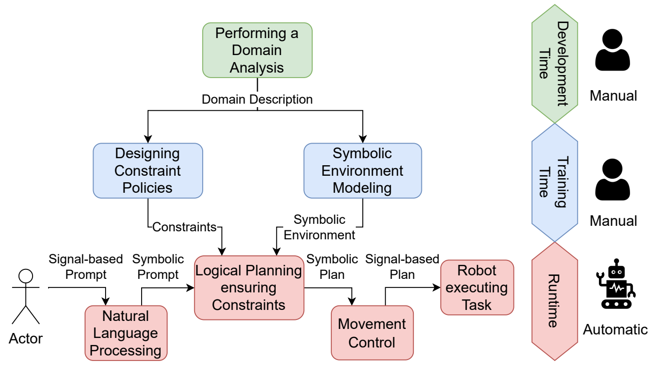

Figure 2: Workflow of designing a robot task planner using logical planning and logical constraints. We differentiate between three phases: In the development phase, the rules for the constraints are designed and the modeling for the logical planner is done. The training phase consists of the incorporation of such constraints into the planner, while during runtime, the plan is generated and executed.

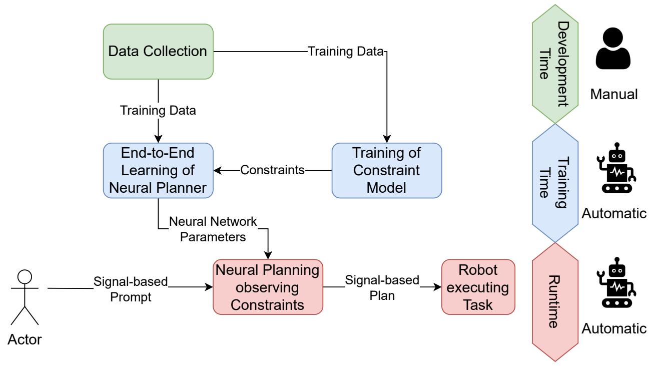

Figure 3: Workflow of designing a neural planner that satisfies neural constraints. The development phase now consists only of data collection. The training phase includes the actual training of the deep neural network, while the neural constraints are being fed into the training of the planner. During runtime, a forward pass through the network is performed.

2 ANALYSIS OF COMBINATIONS

We describe the problem as follows. A prompt is an instruction given by a user to perform a task to reach a certain goal. A signal-based prompt is, e.g., a user-written text or a microphone recording. A symbolic prompt refers to abstract symbols, like subjects, objects, and locations with attributes. A plan is a sequence of actions to be performed by the robot to fulfill the task and reach the goal. A symbolic plan is a sequence of abstract actions with their pre- and postconditions (effects) that is converted to a signal-based plan in the form of actuator commands needed for execution. Planning is the problem of finding such a sequence of actions. Logical planning (usually called symbolic planning) does so by reshaping logical formulas, and needs a symbolic prompt as input to output a symbolic plan. Neural planning is the forward pass through a deep neural network that learned to plan, and can either use a symbolic or signal-based prompt, to either create a symbolic or signal-based plan. End-to-end planning infers a signal-based plan from a signal-based prompt. Constraints are machine-executable representations of policies that have to be observed by a plan execution monitor. A logical constraint is a constraint in form of, e.g., a predicate-logical formula, as in classical access control. A neural constraint is a constraint in form of a deep neural network.

Logical Planning with Logical Constraints. In logical planning, a planner finds a symbolic plan by searching through possible robot actions, trying to find transitions that transform the current state of the robot and its environment to the goal state inferred from the user prompt. In practice, the de-facto standard for this is the Planning Domain Definition Language (PDDL) [3], which we used in the following. Each action has preconditions that need to hold in order for it to be executed. Typical preconditions are that the robot has to be at a table to grab an item and thus its hand should be empty, or that an item has to be unobstructed to be grabbed. Incorporating safety, security, and privacy constraints in form of preconditions is a natural approach to generate plans that are aware of such constraints, as shown in the following.

Since the abstraction of a PDDL transition corresponds to ACAC's main abstraction of an activity, we propose combining these two concepts. This workflow is illustrated in Fig. 2. We identify three different phases when designing such planner. Only during runtime we automatically generate plans, while during development and training time manual work needs to be done. We were able to map[1] the preconditions of ACAC directly into the preconditions of PDDL, as well as the resulting conditions of an activity, which could be represented as *effect* in PDDL.

However, ACAC's contextual conditions and current conditions, which are checked during the execution of an action, are not directly translatable. Contextual conditions can be incorporated using extensions to PDDL, since they rely on querying some external data. Current conditions interfere with PDDL's concept of executing an action instantly that has all its preconditions fulfilled. While it is possible to incorporate toy examples of ACAC into PDDL to some extent, we question the scalability up to the humanoid robots in a care home use case without requiring an equally complex manual specification. Every possible action, every possible object and every ACAC policy needs to be considered, inherently prohibiting such systems from being task-universal.

Neural Planning with Neural Constraints. Recent developments in the field of machine-learning-based access control showed the feasibility of access decisions performed by machine learning methods such as neural networks, as done by Nobi et al. [5]. However, current systems show quite large error margins, reaching a precision of about 0.9. Nobi et al. propose the *Deep Learning Based Access Control* model, where a neural network directly performs access decisions. Such systems are mostly trained on access logs and further user and resource metadata. Note that the neural planning and neural constraints are learned using the same input. We can see such a workflow illustrated in Fig. 3. Again we identify three different phases. Compared to logical planning with logical constraints, we can now perform the training phase automatically, while only the development needs manual work, which improves scalability of solvable problem complexity.

We tested this approach[1] by formulating our running example as natural-language symbolic prompt, and used ChatGPT 3.5 as neural constraint-observing planner to generate natural-language

---

[1] Full artifacts available: https://github.com/kit-dsn/how-to-raise-a-robot-beyond-ac

symbolic plans, shown in Listing 1. Lines 1 and 12 reduce ChatGPT to a neural constraint-observing planner for symbolic prompts and symbolic plans. Lines 2-8 simulate sensors and memory, lines 9-11 instruct ChatGPT with a goal and constraints. We obtain a plan that reaches the goal of cleaning the table, while not interacting with the most personal item on the table, the diary. This experiment shows that we can use a deep-learned neural network to do neural planning while observing neural constraints inferred from symbolic prompts. While the dishes were always moved to the kitchen and the diary was always considered too personal to interact with, the newspaper was sometimes interacted with and sometimes not, showing probabilistic instabilities can occur in this approach.

```
1   Suppose you are a symbolic planning system that generates plans for task–
        universal assistive and humanoid robots based on a domain description
        that is provided below.
2   Domain:
3   – Care home that uses robots to care for their residents. The care home has lots
        of personal rooms for the residents, common spaces like a dining area,
        and administrative spaces like a kitchen or offices
4   – Personal Room of a resident that has a table inside
5   – on the table are the following things:
6     – newspaper that was paid for by the resident
7     – dirty dishes that are owned by the care home and are cleaned by the care
        home
8     – a diary that most certainly contains very private information
9   Goal: Remove as many objects from the table as possible
10  Constraints: Do not interact with objects that you would consider personal.
11  Possible actions: Move, analyze an object, take an object, place an object
12  Put out the plan in terms of a sequence of actions the robot performs. Put out
        the action you used and the object you interacted with or where the
        object was placed. You can only use the actions from the list above.
        Only put out the sequence of actions as formatted list. The plan must be
        conform with all given constraints.
13  # Answer
14  1. Analyze the objects on the table.
15  2. Identify the newspaper as an object that belongs to the resident and should
        not be interacted with.
16  3. Take the dirty dishes from the table and place them in the kitchen.
17  4. Analyze the diary and identify it as an object that contains private
        information and should not be interacted with.
18  5. Move any remaining objects on the table to a different location in the
        personal room where they do not obstruct the table.
```

Listing 1: A prompt for ChatGPT to generate symbolic plans that are learned from a basic domain description plus a ChatGPT answer

While this technique requires less human specification to scale up to higher problem complexity than using logical planning and logical constraints in combination, changes in the policy make retraining of the neural planner and neural constraints necessary. However, ways to tackle policy changes in a manageable way have been shown regarding DLBAC administration [4]. One could also consider incorporating neural constraints into the reward calculation of reinforcement-learning-based planners.

Neural Planning with Logical Constraints. Another possible combination of activity control and robot task planning is presented by combining learned planning with rule-based activity control. Analogously to the previous paragraph, but unconventionally for Generative Adversarial Network (GANs), we can use logical constraints as discriminators to train the end-to-end learning-based neural planners. Although this approach might not be as scalable to complex deployments due to manual policy engineering, it is easier to administrate since the logical constraints do not need to be retrained after a change. For our running example, we now need to incorporate policies on what are personal items again, which makes the whole process inherently more complex. Even though the neural planner learns its behavior, extensive policy engineering needs to be done. Once the training of the end-to-end learning-based planner is finished, this approach behaves similarly to using a learning-based activity control system: Given a prompt, the robot performs a forward pass through the neural planning network and receives a plan that satisfies the desired constraints.

Logical Planning with Neural Constraints. Integrating neural constraints with logical planning can be done in the same way as logical constraints. While this alleviates extensive policy engineering, disadvantages regarding policy administration are retained.

Hybrid Approach. While logical and neural constraints are quite different compared to each other, the integration in logical and neural planning is similar. Therefore, we are not limited to choosing only one combination. We instead argue for a hybrid approach to combine the best of both worlds: one can use neural planning and neural constraints for versatility and universality, but ensure critical policies as logical constraints during runtime. For critical situations with limited universality, one can also fall back to logical planning to find an accurate solution slowly. Neural systems could also serve as recommendation engine for new policies.

3 CONCLUSION

We discussed combinations of logical and neural task planning and activity control approaches. We highlighted their trade-offs and showed their benefits and shortcomings. While neural planners scale better to complex deployments in terms of required manual specification, they introduce an error margin and hinder administrability. Designing logical planning systems, however, requires more manual tasks than neural planning systems. In order to create deployable systems one should, therefore, strive for hybrid designs to combine best of both worlds: neural planning with neural constraints leads to probable observance of policies, critical policies will be safeguarded by logical constraints. As next step, we will test the feasibility of such a hybrid approach in real robot experiments.

Acknowledgements. This work has been supported by the project "Stay young with robots" (JuBot). The JuBot project was made possible by funding from the Carl Zeiss Foundation.

REFERENCES

[1] Saskia Bayreuther, Florian Jacob, Markus Grotz, Rainer Kartmann, Fabian Peller-Konrad, Fabian Paus, Hannes Hartenstein, and Tamim Asfour. 2022. BlueSky: Combining Task Planning and Activity-Centric Access Control for Assistive Humanoid Robots. In *Proceedings of the 27th ACM Symposium on Access Control Models and Technologies (SACMAT '22)*. ACM, New York, 185–194. https://doi.org/10.1145/3532105.3535018

[2] Maanak Gupta and Ravi Sandhu. 2021. Towards Activity-Centric Access Control for Smart Collaborative Ecosystems. In *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies (SACMAT '21)*. ACM, New York, 155–164. https://doi.org/10.1145/3450569.3463559

[3] Adele Howe, Craig Knoblock, Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. 1998. *PDDL – The Planning Domain Definition Language.* Technical Report. Yale Center for Computational Vision and Control.

[4] Mohammad Nur Nobi, Ram Krishnan, Yufei Huang, and Ravi Sandhu. 2022. Administration of Machine Learning Based Access Control. In *ESORICS 2022*. Springer, Cham, 189–210. https://doi.org/10.1007/978-3-031-17146-8_10

[5] Mohammad Nur Nobi, Ram Krishnan, Yufei Huang, Mehrnoosh Shakarami, and Ravi Sandhu. 2022. Toward Deep Learning Based Access Control. In *Proceedings of the Twelfth ACM Conference on Data and Application Security and Privacy (CODASPY '22)*. ACM, New York, 143–154. https://doi.org/10.1145/3508398.3511497