Proactive Learning of Cognitive Exercises with a Social Robot*

Alejandro Suárez-Hernández[1] and Antonio Andriella[2] and Guillem Alenyà[1] and Carme Torras[1]

***Abstract*— We introduce *INtuitive PROgramming 2* (INPRO2), an improvement over our previous INPRO framework for learning board exercises via demonstrations. INPRO2 makes use of our *Online Action Recognition through Unification* (OARU) algorithm, which maintains and extends as needed a library of STRIPS action schemata that represent the dynamics, rules and goal of the exercise. OARU operates on a sequence of states shown by the user. Each state transition is either used to learn a new action, or is recognized as an instance of one action currently present in the library, possibly refining it. We have extended OARU to support negative examples (i.e. invalid moves that show forbidden state transitions) in order to increase the complexity of the exercises that can be learned. This new OARU's feature is exploited through another crucial element of INPRO2: its ability to proactively ask for the legality of certain moves to the user in critical situations, and fix overly permissive actions. We show an example of a typical INPRO2 learning session. We also outline a plan for a user study that will serve to assess the proactive behavior of the robot.**

I. INTRODUCTION

The field of *Socially Assistive Robotics* (SAR) studies the applications of robotics to help humans through communication and guidance, rather than physical means. One such application is delivering cognitive training therapies to patients affected by cognitive impairments (e.g. Alzheimer's or dementia) via board exercises [1].

To this end, Andriella *et al.* [2] designed a socially assistive robot capable of providing in situ personalised assistance to Persons with Dementia (PwDs) during their daily cognitive training therapy. However, one limitation of this approach was that it did not allow the therapists to change exercises in an easy and intuitive way. Indeed, the most straightforward approach for implementing such a system relies on a hand-crafted description of the rules of the exercise. In this scenario, technical skills beyond the competence of a healthcare professional are needed to add new exercises to the robot's repertoire. This obstacle can be circumvented allowing the robot to learn new exercises via demonstrations provided by the caregiver. In the past, we have tackled this robot-caregiver interaction, and proposed the *INtuitive PROgramming* (INPRO) framework [3], [4] for learning STRIPS [5] action schemata that encode the rules of the exercise in their preconditions. We now introduce INPRO2, which features significant improvements over INPRO. Fig. 1 depicts a learning session of INPRO2.

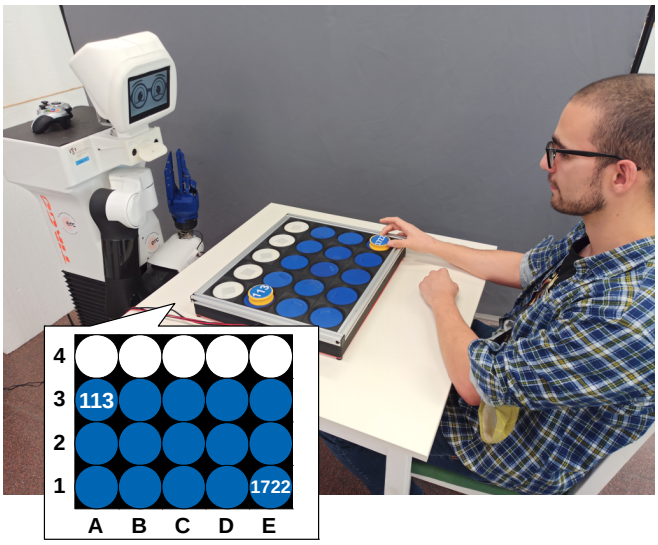

Fig. 1. Teaching an exercise consisting in moving the odd numbers horizontally to the right column, and the even numbers vertically to the top row, always one cell at a time (i.e. tokens must move to adjacent cells).

Unlike INPRO, INPRO2 relies on *Online Action Recognition through Unification* (OARU) [6], an algorithm for learning and recognizing STRIPS action schemata from a stream of states given by the user (the caregiver, in this case). While OARU was conceived to work under partial observability (in particular, with the open world assumption), in the present work we focus on the fully observable setting.

Our contributions are: **(1)** the use of OARU in INPRO2 allows the robot to learn much more complex exercises than INPRO; **(2)** we have augmented OARU with the ability to learn from forbidden state transitions (negative examples) to fix overly permissive preconditions; **(3)** we have enabled INPRO2 to learn goals, in addition to legal moves; and **(4)** INPRO2 proactively intervenes in critical moments of the teaching process, asking for the legality of certain actions to potentially discover negative examples and learn from them.

II. RELATED WORK

Martínez *et al.* use *Reinforcement Learning* (RL) to learn actions with stochastic effects [7] with the help of teacher demonstrations. Unlike INPRO2, this approach requires action signatures (i.e. the number of available actions and their parameters) to be given. *MuZero* [8] has been used to learn games like chess and go. Notwithstanding its ability to learn

*The research leading to these results receives funding from the EU H2020 Programme under grant agreement ERC–2016–ADG–741930 (CLOTHILDE); ASH acknowledges the financial support of the Apadrina la Ciencia association and of the Ford Spanish branch.

[1]Authors are with Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Parc Tecnològic de Barcelona. C/ Llorens i Artigas 4-6, 08028, Barcelona, Spain {asuarez, torras, galenya}@iri.upc.edu

[2] A. Andriella is with Pal Robotics, C/ de Pujades, 77, 08005 Barcelona, Spain. antonio.andriella@pal-robotics.com

both the rules of the game and play skillfully, it is not feasible for a caregiver to provide the amount of required data.

Inverse RL (IRL) [9] makes use of the teacher demonstrations to learn the reward of the different actions given the current state. This can be used in principle to distinguish between legal and illegal moves (actions with high and low reward, respectively). While this does not require such an exhaustive amount of training data as MuZero, it does not provide precise action descriptions (e.g. in STRIPS format), and thus lack INPRO and INPRO2's explainability.

Algorithms for inductive learning of high-level action models [10], [11], [12] observe interactions with the environment, and refine planning operators to match the observations. System-centric algorithms like ARMS [13] generate deterministic planning operators with weighted MAX-SAT solvers. SLAF [14] learns from partial observations. FAMA [15] computes STRIPS operators from minimal observations using classical planning. UDAM [16] uses a similar approach, but removes the limitation of requiring action signatures. INPRO [4] was inspired by the latter two works. However, INPRO2 uses OARU [6], which is based on a notably different principle of clustering actions more suitable for online learning and recognition.

The motivation of Senft *et al.* [17] and Efthymiou *et al.* is reminiscent of ours. In particular, they empower adult educators to design the behavior of a robotic tutor for children. The work of Winkle *et al.* [18] seeks to increase the involvement of medical personnel in tuning the interaction of SAR robots with patients. These methods, while adequate for learning how to interact, are not meant to learn precise logical descriptions of board exercises.

III. PRELIMINARIES

To properly explain INPRO2, let us first introduce some concepts from First-Order Logic (FOL) and STRIPS.

We denote as $\mathcal{D}$ the *domain of discourse*, a set of world objects. In Fig. 1, for instance, board locations *a1* and *b2*, and tokens *113* and *1722* are objects within $\mathcal{D}$. A *predicate* consists of a predicate symbol p and certain *arity* n. A *predicate variable* is an n-ary predicate parameterized with a list of n objects (e.g. $p(x_1, \ldots, x_n)$, with $\{x_1, \ldots, x_n\} \subseteq \mathcal{D}$), and evaluates to either *true* or *false*. An interpretation of an n-ary predicate is a set of n-tuples from $\mathcal{D}^n$ for which a predicate evaluates to *true*. In a slight abuse of notation, from now on we will refer to predicate variables simply as predicates. In FOL, we use *formulas* to express statements over the objects in $\mathcal{D}$. In this paper, we consider restricted formulas that consist of either single predicates or conjunctions ($\wedge$) of predicates[1]. Consider, for instance, the predicates $at(x, y)$ and $empty(z)$, whose semantics are, respectively, "token *x* is at cell *y*" and "cell *z* is empty". In Fig. 1, the following formulas evaluate to *true*:

- $at(113, a3)$
- $at(1722, e1) \wedge empty(e4)$;

[1]Of course, the full specification of FOL includes negations, disjunctions and quantifiers, but these are not considered here.

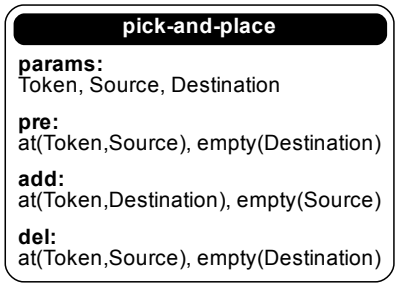

Fig. 2. Pick and Place action schema

while the following ones evaluate to *false*:

- $empty(a3)$;
- $at(113, a3) \wedge at(1722, a1)$;

A *state* s consists of a collection of assignments to a set of binary variables. Alternatively, s can be viewed as an interpretation over a set of predicates[2], each one representing a fact about the world. A state can be compactly represented as the set of *active predicates* (i.e. predicates that evaluate to *true*). The following is an excerpt of the state from Fig. 1:

$$s = \{at(113, a3), at(1722, e1), empty(a4), empty(b4), \ldots\}$$

STRIPS is a formalism to specify actions. A STRIPS action schema is a tuple $a = \langle head_a, pre_a, add_a, del_a \rangle$, where:

- $head_a$ is the *action signature*. It consists in a duad $\langle N_a, P_a \rangle$, where N_a is a human-readable name for the action, and P_a is a list of free variables which constitute the action parameters that take values over $\mathcal{D}$.
- pre_a is the *precondition*, a restricted FOL formula that must evaluate to true in the current state in order for the action to be applicable.
- add_a is the *add list*, the list of predicates that will be set to *true* (or added to the compact form of the state) after the execution of the action.
- del_a is the *delete list*, the list of predicates that will be set to *false* (or deleted from the compact form of the state) after the execution of the action.

Predicates in pre_a, add_a, and del_a may be parameterized with objects and free variables from $D \cup P_a$. The action is grounded when every parameter in P_a takes a value from D. Fig. 2 shows an example of a STRIPS action schema. Such action allows the move of any token to any empty position of the board. As a convention in this paper, names starting with uppercase letters are action parameters.

IV. METHODOLOGY

When INPRO2 starts learning a new exercise, it starts with an empty action library and follows the next flow:

1) Before receiving demonstrations, the tokens available for the exercise are specified by the user (this is to establish the domain of discourse $\mathcal{D}$).
2) The initial position for a new run of the exercise is set.

[2]Since predicates are statically evaluated, in dynamical domains the concept of *fluent* (facts that vary over time) is often used instead. Despite this technicality, for simplicity we will adhere to the use of the term predicate.

3) INPRO2 manipulates its internal library of learned STRIPS action schemata to accommodate the demonstrations made by the user.
4) If certain criteria are met, INPRO2 asks for the legality of a move different from the user's. If the move is illegal, the action library is refactored to disallow it.
5) Once the user has finished demonstrating one run of the exercise, they either conclude the teaching process or go back to 2 to show another run of the exercise. At this point, INPRO2 uses the last state of the run to learn or refine the goal of the exercise.

For each each run R_i of the exercise, INPRO2 receives a stream of states $R_i = \{s_{i1}, s_{i2}, \ldots, s_{il_i}\}$, where l_i is the length of the run. We denote each pair of consecutive states $(s_{ij}, s_{i(j+1)})$ as observation o_{ij}. Internally, INPRO2 uses the OARU algorithm to process each observation. In particular, for each observation OARU may: (1) simply recognize the transition as the result of a grounding of one of the actions in its library; (2) introduce parameters and relax the precondition of one of the actions in its library to allow the observed transition; or (3) add an entirely new action with a stringent precondition, with the intention of relaxing it when future similar transitions are observed.

OARU does not require the action signatures to be given beforehand, as it builds its action library in an online fashion. It merges together similar actions through a process called *Action Unification* (AU). To merge two actions, AU introduces parameters and relaxes preconditions. Therefore, AU can be seen as a tool for generalizing actions. OARU uses AU as a subroutine to perform hierarchical clustering over the learned actions. The details of this approach can be seen in the original paper [6].

One of the original limitations of OARU is that it always merges actions with equal effect. This is undesirable when the only correct way of accurately reflecting the rules of an exercise using STRIPS is with two actions with identical effects but different preconditions. We overcome this by enhancing OARU with the ability to undo previously merged actions via *negative examples* (forbidden transitions). Whenever OARU registers a new negative example, it refactors its action library so none of the actions can produce the shown transition. Fig. 3 depicts this mechanism. Like in Fig. 1, the odd numbers are meant to move one cell at a time to the right and the even ones one cell at a time to the top. Since these two kinds of move have the same effect, OARU merges them together into a single generic "pick and place". However, after just one negative example, OARU is able to fix this mistake. This feature allows OARU to learn much more complex domain mechanics than before.

Another important feature of INPRO2 is its ability to learn goals of moderate complexity. This has been achieved by appending a new state $s_{i(l_i+1)} = s_{il_i} \cup \{goal\text{-}achieved()\}$ at the end of each run R_i. Effectively, this leads OARU to learn a special action to assert the achievement of the goal. The only effect this special action has on the state is setting the *goal-achieved*() predicate to *true* and, once learned correctly, the precondition reflects the goal of the exercise.

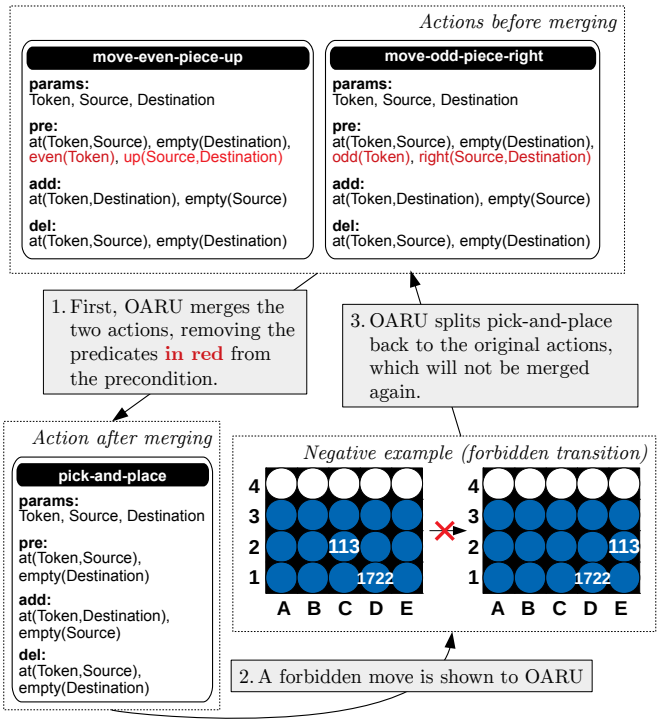

Fig. 3. Undoing an overly relaxed action thanks to a negative example. The human-readable names of actions and parameters have been chosen by us for the purpose of clarity in this paper, not by INPRO2.

To elicit and exploit negative examples, we take advantage of the learned goals. Actions that are learned incorrectly usually have overly relaxed preconditions. Therefore, INPRO2 may find plans towards the goal significantly shorter than the length of the demonstrated runs. When the user demonstrates a move that does not agree with the optimal move found by INPRO2, this is an indicative that one of the learned actions is too lax. The robot, then, asks the user about the legality of alternative moves and gather potential negative examples. In addition, INPRO2 periodically asks about the legality of alternative moves to gather negative examples in situations when the overly relaxed actions do not allow shorter plans.

Fig. 4 shows an example execution, with two runs. During the first run, R_1, INPRO2 learns a generic, and incorrect, pick and place action schema, like the one from Fig. 3. It also learns the goal of the game: the token *113* must be in the right column, while *1722* must be in the top row. During the second run, R_2, after the first move of the user, the robot asks why they did not move the token *1722* directly to the top row. After the user declares that that is illegal, INPRO2 splits the pick and place action into two correct actions for moving separately the odd and even pieces. In the bottom right corner we show the final set of learned actions.

V. PLANNED EXPERIMENTS

We pose the following research questions:

- **(R1)** Does our selective intervention criteria harm learning speed?
- **(R2)** Does our selective intervention criteria result in a better user experience than more frequent interventions?

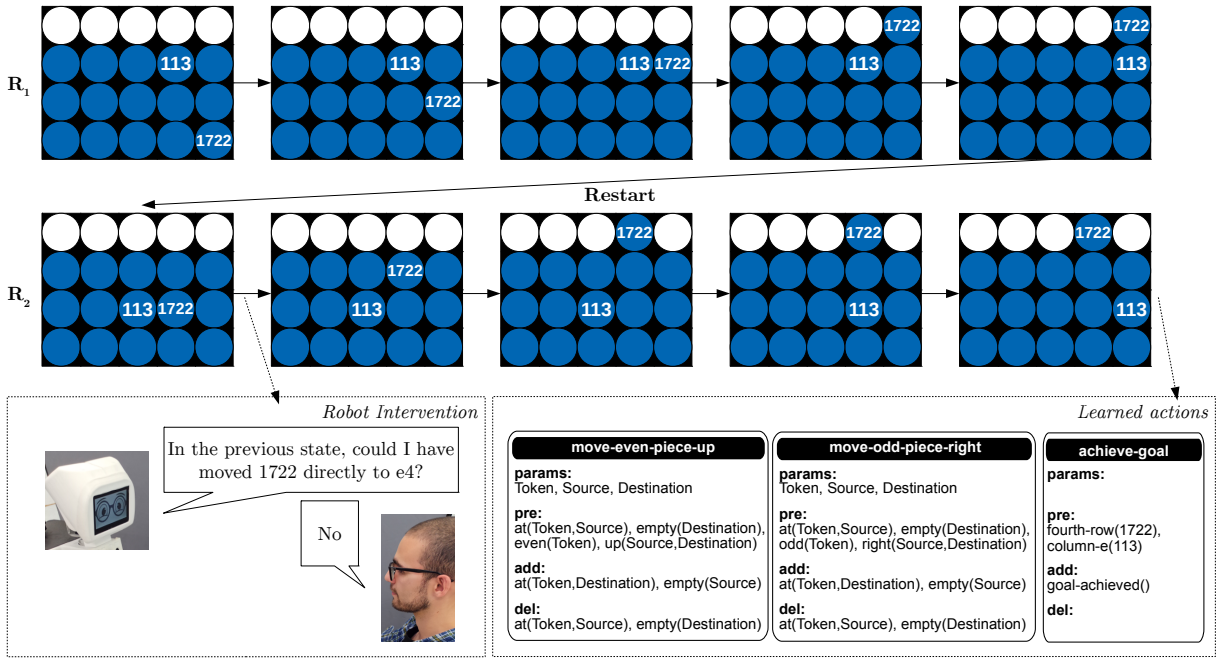

Fig. 4. Execution example: teaching the robot to move odd numbers to the right and even numbers to the top of the board.

We hypothesize that the answer to these questions is *no* and *yes*, respectively.

We plan to perform a user study with several participants with diverse backgrounds. We will give the participant a brief tutorial on the usage of INPRO2, and then we will ask them to teach the robot two different exercises. We assume that all the demonstrations made by the user are correct, so we will ask them to undo moves if they commit a mistake. The participants will be split into three groups: (1) one with occasional, planned robot interventions as explained in this paper; (2) one with occasional robot interventions, but in random moments; and (3) one final group with exhaustive questioning after every move. We will measure how many demonstrations it takes before the robot learns the rules of each exercise, and after each experiment, we will ask the participants to fill a series of user experience and cognitive load questionnaires. In the future, we also plan to compare INPRO2 with IRL in simulation.

References

[1] A. Andriella, C. Torras, and G. Alenyà, "Cognitive System Framework for Brain-Training Exercise Based on Human-Robot Interaction," *Cognitive Computation*, vol. 12, no. 4, pp. 793–810, 7 2020.

[2] A. Andriella, C. Torras, C. Abdelnour, and G. Alenyà, "Introducing CARESSER: A framework for in situ learning robot social assistance from expert knowledge and demonstrations," *User Modeling and User-Adapted Interaction*, 3 2022.

[3] A. Andriella, A. Suárez-Hernández, J. Segovia-Aguas, C. Torras, and G. Alenyà, "Natural teaching of robot-assisted rearranging exercises for cognitive training,," in *Social Robotics*. Springer International Publishing, 2019, pp. 611–621.

[4] A. Suárez-Hernández, A. Andriella, A. Taranović, J. Segovia-Aguas, C. Torras, and G. Alenyà, "Automatic learning of cognitive exercises for socially assistive robotics," in *2021 30th IEEE International Conference on Robot Human Interactive Communication (RO-MAN)*, no. 1, 2021, pp. 139–146.

[5] R. E. Fikes and N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, 1971.

[6] A. Suárez-Hernández, J. Segovia-Aguas, C. Torras, and G. Alenyà, "Online Action Recognition," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 13, pp. 11 981–11 989, 2021.

[7] D. Martínez, G. Alenyà, and C. Torras, "Relational reinforcement learning with guided demonstrations," *Artificial Intelligence*, vol. 247, pp. 295–312, 6 2017.

[8] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver, "Mastering Atari, Go, chess and shogi by planning with a learned model," *Nature*, vol. 588, no. 7839, pp. 604–609, 12 2020.

[9] S. Arora and P. Doshi, "A survey of inverse reinforcement learning: Challenges, methods and progress," *Artificial Intelligence*, vol. 297, p. 103500, 8 2021.

[10] Y. Gil, "Learning by Experimentation: Incremental Refinement of Incomplete Planning Domains," in *Machine Learning Proceedings 1994*. Elsevier, 1994, pp. 87–95.

[11] S. Benson, "Inductive Learning of Reactive Action Models," in *Machine Learning Proceedings 1995*. Elsevier, 1995, pp. 47–54.

[12] X. Wang, "Learning by Observation and Practice: An Incremental Approach for Planning Operator Acquisition," in *Machine Learning Proceedings 1995*. Elsevier, 1995, pp. 549–557.

[13] Q. Yang, K. Wu, and Y. Jiang, "Learning action models from plan examples using weighted MAX-SAT," *Artificial Intelligence*, vol. 171, no. 2-3, pp. 107–143, 2007.

[14] E. Amir and A. Chang, "Learning Partially Observable Deterministic Action Models," *Journal of Artificial Intelligence Research*, vol. 33, pp. 349–402, 11 2008.

[15] D. Aineto, S. Jiménez, and E. Onaindia, "Learning STRIPS Action Models with Classical Planning," *ICAPS*, 2018.

[16] A. Suárez-Hernández, J. Segovia-Aguas, C. Torras, and G. Alenyà, "STRIPS Action Discovery," in *AAAI 2020 workshop on Generalization in Planning*. arXiv, 2020.

[17] E. Senft, S. Lemaignan, M. Bartlett, P. Baxter, and T. Belpaeme, "Robots in the classroom: Learning to be a Good Tutor," in *Robots for Learning Workshop at HRI 2018*, 2018.

[18] K. Winkle, S. Lemaignan, P. Caleb-Solly, P. Bremner, A. Turton, and U. Leonards, "In-Situ Learning from a Domain Expert for Real World Socially Assistive Robot Deployment," in *Robotics: Science and Systems XVI*, vol. 10. Robotics: Science and Systems Foundation, 7 2020.