# Automatic Learning of Cognitive Exercises for Socially Assistive Robotics

[1]Alejandro Suárez-Hernández, [1]Antonio Andriella, [3]Aleksandar Taranović,
[2]Javier Segovia-Aguas, [1]Carme Torras, [1]Guillem Alenyà

***Abstract*— In this paper, we present a learning approach to facilitate the teaching of new board exercises to assistive robotic systems. We formulate the problem as the learning of action models using Boolean predicates, disjunctive preconditions, and existential quantifiers from demonstrations of successful exercise executions. To be able to cope with exercises whose rules depend on a set of features that are initialized at the beginning of each play-out, we introduce the concept of *dynamic context*. Furthermore, we show how the learnt knowledge can be represented intuitively in a graphical interface that helps the caregiver understand what the system has learnt. As validation, we conducted a user study in which we evaluated whether and to which extent different types of feedback can affect the subjects' performance while teaching three types of exercises: (1) sorting *numbers*; (2) arranging *letters*; and (3) reproducing *shapes* sequences in reversed order. The results suggest that textual and graphical feedback are beneficial.**

## I. Introduction

Socially Assistive Robotics (SAR) is a field of robotics that provides assistance through social rather than physical interaction [1]. SAR has been shown to be a powerful tool to assist caregivers in cognitive training, engaging and evaluating mild dementia patients [2]. However, one of the main limitations is that they are not usually developed to be easily re-programmed [3] [4]. Drawing inspiration from recent work on learning action models [5] and high-level features for classical planning [6], we have recently explored the idea of rule learning for simple games [7].

In this paper we present the **INPRO** (**In**tuitive **Pro**gramming) learning framework. Unlike inverse reinforcement techniques [8], INPRO uses much less data to infer the exercise's rules. Moreover, INPRO is much more explainable since rules are obtained as first-order logic descriptions that can be easily translated to textual and graphical feedback. INPRO's purpose is enabling the caregiver to program a robot so it can play and monitor cognitive exercises, explaining the actions that are applicable in every situation. INPRO learns from *execution traces*, and each trace represents a sequence of pick-and-place actions and states demonstrated by the user (Fig. 1). The learning task is compiled to a classical planning problem in PDDL (Planning Domain Definition Language), and then is solved using a SAT planner. This combination allows us to extract simple, precise logical descriptions of the exercise's actions. Our contributions in this paper are:

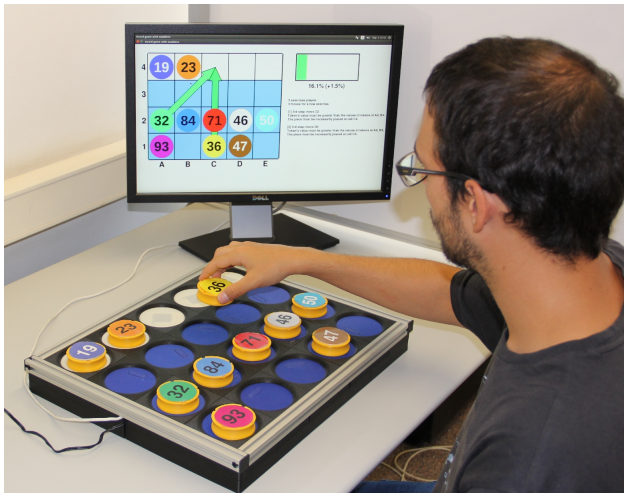


Fig. 1: A participant teaching a new game using the board and receiving feedback in the screen.

- **Context:** Rules are expressed as logic formulas that contain boolean predicates or descriptors. We introduce here the notions of *static* and *dynamic context*. The first does not change from play-out to play-out, while the second are useful to define exercises that require some sort of initialization that is not necessarily the same in each play-out (e.g. memory exercises).
- **Intuitive feedback:** the representation allows to provide natural language description of the rules, and also a *graphical representation of the rules* and a *learning indicator*. This feedback is meant to help the caregiver decide if the system has fully understood the exercise or needs more traces to infer the rest of the rules.
- **User study:** We present empirical support for our framework in the form of a *user study* on friendly explanations for intuitive programming.

Inspired by the Syndrom Kurztest (SKT) [9], we have designed three different exercises to train patients' cognitive abilities, namely their memory, attention, and visuomotor functions [10]. The exercises require: (1) ordering *numbers*; (2) arranging *letters*; (3) and reproducing sequences of

A. Andriella, C. Torras and A. Suárez-Hernández were partially funded by the European Union´s Horizon 2020 under ERC Advanced Grant CLOTHILDE (no. 741930), G. Alenyà by the EU H2020 research and innovation programme IMAGINE (no. 731761) and J. Segovia-Aguas by the programme TAILOR (no. 952215). The work was partially supported by the Spanish State Research Agency through the María de Maeztu Seal of Excellence to IRI (MDM-2016-0656).

[1] A. Suárez-Hernández, A. Andriella, C. Torras and G. Alenyà are with Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Llorens i Artigas 4-6, 08028 Barcelona, Spain. {asuarez, aandriella, torras, galenya}@iri.upc.edu

[2] J. Segovia-Aguas is with Pompeu Fabra University, Tànger, 122-140 08018, Barcelona, Spain.{javier.segovia@upf.edu}

[3] A. Taranović is with Bosch Center for Artificial Intelligence, Renningen, Germany.{aleksandar.taranovic@de.bosch.com}

*shapes* in reverse order. They were implemented on a board with *10* tokens and a screen to provide the different feedback modalities (see Fig. 1). Two of the exercises (1 and 2) can be solved with only the static context (order relations between numbers and letter-to-location correspondences). On the other hand, in exercise 3, the sequence is meant to be memorized by the patient at the beginning and is bound to change after each completion of the exercise. Therefore, it requires *dynamic context* or knowledge that is instance dependant.

We have also conducted a user study to evaluate the performance of two groups of participants: (1) one teaching with minimum feedback (only an indicator of whether the system has learnt something or not from the last execution); and (2) another that uses all the feedback types (text, graphical assistance, and learning indicator).

## II. Related Work

One of the challenges we propose consists in making assistive robots adaptive to learn new cognitive exercises. While typically applied to situated agents, Reinforcement Learning (RL) has been also used to learn probabilistic action models [11], even in the presence of exogenous effects [12]. However, these techniques are inherently heuristic and meant for non-deterministic environments.

Transfer learning [13] and Inverse RL [8] are two techniques that can be applied to make the learning process adaptable and scalable. While the first one uses previous knowledge to aid in solving new tasks and adapting the knowledge representation to new situations, the latter learns from demonstrations by a teacher. Contrarily to INPRO, these techniques cannot explain the intrinsic rules that describe the exercise.

Inductive learning of high-level action models consists of incremental approaches to acquire knowledge [14]–[16] where agents are observed interacting with the environment and planning operators are refined to meet the observations. *System-centric* algorithms such as ARMS [17] can generate deterministic planning operators with weighted MAX-SAT solvers. SLAF [18] can learn from partial observations. OARU [19] is substantially different from the former, in that it follows a clustering approach to generalize actions. FAMA [5] can compute STRIPS operators from minimal observations, i.e. a set of planning instances each with initial state and goal condition. Inspired by the recent success of the later, we force the system to satisfy all clauses using a SAT-based solver with the intention of learning inductively the rules of exercises.

Senft *etal* [20]'s have a very similar motivation to us: they seek to enable robots to learn from adult teachers how to tutor children in a classroom. Also similar in scope, we find the work of Winkle *etal* [21] for assisting medical personnel. However, these methods, while adequate for learning how to interact, are not so fit for learning precise logical descriptions of board exercises.

## III. Preliminaries

Internally, INPRO learns a set of action models that are in the form of PDDL (Planning Domain Definition Language) schemas. INPRO works by compiling the learning task itself to a classical planning problem in PDDL. Since both INPRO's inner operation and output are influenced by classical planning and action model learning, this section will give a brief overview of these topics.

### A. Classical Planning

Classical Planning is characterized by search problems in deterministic, fully-observable and non-concurrent environments. Formally, a *classical planning problem* [22] is defined by a 4-tuple $P = \langle F, A, I, G \rangle$ , where $F$ is a set of fluents or predicates, $A$ is a set of actions, $I \subseteq F$ is the initial state of the problem and $G \subseteq F$ is the goal condition.

The set of predicates in a planning problem $P$ implicitly defines a state space of $S = 2^F$. The states follow the close-world assumption, where predicates whose value is not indicated are considered false. Therefore, a state can be compactly specified by the set of fluents that are set to true. The set of goal states is $S_G = \{s \in S | G \subseteq s\}$ or, in other words, all the states where the goal condition holds. An action $a \in A$ consist of 3 sets of fluents: (i) the precondition $\mathsf{pre}(a) \subseteq F$, (ii) the negative effect $\mathsf{del}(a) \subseteq F$, and (iii) the positive effect $\mathsf{add}(a) \subseteq F$. An action $a$ is *applicable* to state $s$ if and only if the precondition of the action hold in the $s$, i.e. $\mathsf{pre}(a) \subseteq s$. Once an action $a$ is applied, the state transitions according to the following function $\theta(a, s) = (s \backslash \mathsf{del}(a)) \cup \mathsf{add}(a)$. That is, all the predicates in $a$'s negative effect are removed from $s$, while the positive ones are added.

The aim of planning is to compute a plan $\pi$ that consists of a sequence of actions $a_1, \ldots, a_n$. A plan is *sound* if every action is applicable in the corresponding state following the transition function $s_i = \theta(a_i, s_{i-1})$ s.t. $1 \leq i \leq n$, and the goal condition holds in the last state, i.e. $G \subseteq s_n$. Iff a plan $\pi$ is sound, then it is a solution to planning problem $P$.

Classical planning problems can be expressed in PDDL, a declarative language for representing abstract domain dynamics and instances. The full specification of the language is given by McDermott *etal* [23]. While there is no room to detail all the nuances of PDDL, it is worth highlighting that it allows to compactly represent the preconditions and effects of actions thanks to quantifiers, conjunctions, disjunctions, and conditional effects. If preconditions and effects consist exclusively of predicates (i.e. no quantifiers nor conditional effects), it is said that the planning domain has STRIPS expressiveness [24].

### B. Learning Action Models

Actions model can be learnt from the execution of examples or *traces*. A trace, for the purpose of this paper, consists in a history of interleaved states and actions $\tau = \{s_0, a_1, \ldots, a_n, s_n\}$. Algorithms for learning STRIPS action models $\Lambda$ receive as input a full or partial trace $\tau$. Their

objective is computing, for each action $a \in A$, its precondition $\mathsf{pre}(a)$ and effect $\mathsf{eff}(a) = \langle \mathsf{del}(a), \mathsf{add}(a) \rangle$. It is also the case that learners can be aided with partially specified models. This is the case of INPRO, since exercises follow a pick-and-place mechanic and the effect of the actions is fully known beforehand (more on this later).

INPRO uses the STRIPS fragment of PDDL, plus disjunctive preconditions and existential quantifiers. INPRO receives as input a tuple $\Lambda = \langle \mathcal{P}, \mathcal{A}, \mathcal{T} \rangle$, where (1) $\mathcal{P}$ is the set of predicates that describe the board, the relationships between tokens, and the tokens' features; (2) $\mathcal{A}$ is the set of partially specified actions models; and (3) $\mathcal{T}$ is the set of traces. The output is a plan $\pi$ that refines the current action schemas and validates it according to the given traces. The PDDL form of the schemas can be extracted directly from this plan.

The techniques for learning action models are diverse and need to leverage knowledge acquisition with reasoning. Thus, the state-of-the-art in this field depends on the combination of inputs/outputs, and language expressiveness, since this change completely the complexity of each approach [25].

## IV. INTUITIVE PROGRAMMING (INPRO) METHOD

INPRO is an interactive framework in that it learns in real time from exercise demonstrations. INPRO's objective is to find the preconditions of a set of pick-and-place actions with known effects. These preconditions implicitly encode the rules of the exercise.

Our framework is able to quickly infer some of the preconditions, if not all, after a few execution traces. It is up to the caregiver to decide when to stop providing traces. In order to help them take this decision, INPRO's shows its internalized knowledge in several ways: a textual description of the rules, a graphical representation of the applicable actions, and an indicator that tells whether the last demonstrated example was useful to relax the preconditions further.

### A. Overview of the Learning Cycle

INPRO's starts out with a set of maximally restrictive actions and works by progressively relaxing their preconditions watching how they are used in the input traces. To do so, all the traces are transformed into a classical planning problem. The solution of this planning problem is a plan that contains instructions for relaxing the preconditions of the internalized action schemes, and instructions for validating the programmed instructions against the provided traces. Preconditions are expressed in terms of the predicates that describe the status of the board and the context, so they are relaxed by removing predicates from them.

The learning cycle is summarized in Fig. 2. Lines 1 and 2 show the initialization of the algorithm. $A$ is initialized to a set of heavily constrained pick-and-place whose effects are already known (and do not change through the algorithm). $T$, on the other hand, is meant to store the set of input traces, so it is initially empty. At the beginning of each iteration of the main loop, INPRO queries the user for a new demonstration (line 4), that is, a sequence of interleaved states and actions

```
1:  A ← maximally_restrictive_set_of_actions
2:  T ← {} // set of traces
3:  repeat
4:      trace ← gather_new_trace_from_user()
5:      T ← T ∪ {trace}
6:      validated ← try_to_validate_trace(A, trace)
7:      if not validated then
8:          planning_problem ← build_planning_instance(T)
9:          plan ← SAT_planner(planning_problem)
10:         A ← relax_preconditions(A, plan)
11:     end if
12: until caregiver signals end of teaching process
13: output A
```

Fig. 2: Algorithm describing the learning cycle.

that represent a valid play-out. This sequence is added to $T$ (line 5). The system checks if it can already validate the last input trace with its current set of schemas (line 6 and 7). Validating a trace means verifying that the preconditions of the actions that appear in the trace are not so restrictive that they prevent their execution, according to the current's system knowledge. If the trace cannot be validated, a new planning instance is built to find the preconditions' predicates that should be removed to validate all the traces in $T$ (lines 8 and 9). The resulting plan is used to relax the preconditions of the actions in $A$ (line 10) and the loop starts over unless the caregiver deems that the system has already learnt all the rules.

While this algorithm outlines the general learning procedure, there are a few concepts that need a more in-depth explanation. In Section IV-B we analyze how states are described in terms of predicates. This is critical because these predicates are used to specify the actions' precondition. In Section IV-C, we delve into the learnt action representation. More details about the compilation to classical planning are given in Section IV-D.

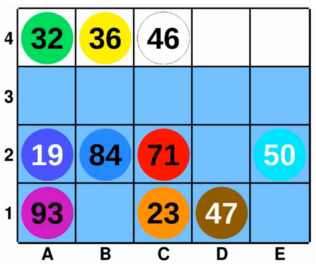
### B. Board Status, Static Context, and Dynamic Context

Exercises are played on a board with dimensions $5 \times 4$. In order to provide traces to INPRO, states have to be represented as sets of predicates. States are composed of: (1) the board status; (2) the static context of the exercise; and (3) the dynamic context of the instance.

**Board status**: Current configuration of the board, which consists of the contents of each cell. The contents are specified through the binary predicate *contains*. The board status changes after the execution of an action, and therefore these predicates are modified as a result, too.

**Static context**: Set of predicates that represent static or immutable properties of the tokens or the cells. These properties are fixed among all the instances of one particular exercise. Examples of such predicates are those that define an order relation, like *lessthan*.

**Dynamic predicates**: Set of predicates that represent some properties that are fixed at the beginning of each exercise instance. While these do not change as the result of the

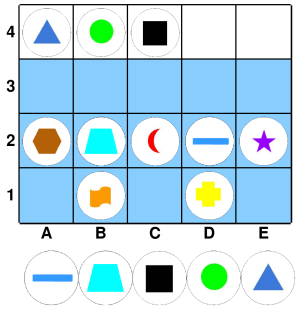



(a)

```
(contains A4 v32)
(contains B4 v36)
...
(contains D1 v47)
(contains E1 empty)
(lessthan v19 v23)
(lessthan v19 v32)
...
(lessthan v46 v47)
(lessthan v47 v50)
```

(b)

Fig. 3: (a) Example configuration of board with numeric tokens. (b) Description of the state represented by the board as predicates. Predicates in regular black color correspond to the status of the board itself, which may change as a result of a move action, while predicates in blue belong to the static context and are fixed among all instances of the same exercise.




(a)

```
(contains A4 triangle)
(contains B4 circle)
...
(contains D1 cross)
(contains E1 empty)
(selected line first)
(selected trapezium
second)
(selected square third)
(selected circle fourth)
(selected triangle fifth)
```

(b)

Fig. 4: (a) Example configuration of a board with shapes. This exercise features a sequence meant to be remembered by the patient, and is shown below for the teacher. (b) Predicates in red belong to the dynamic context and are fixed at the beginning of each trace. The selected tokens and their order may change between consecutive plays.

application of actions, they may potentially be altered when the exercise is restarted. This sort of predicates is useful for teaching exercises whose objective is conditioned by external influence (e.g. a random sequence of tokens meant to be remembered). The *shapes* exercise briefly mentioned in Section I and described in more detail in Section VI uses this sort of context.

Fig. 3 showcases an example of a board and its accompanying description as a set of predicates. The dynamic context in this example is empty. On the other hand, Fig. 4 features a non-empty dynamic context that is used to store a sequence of tokens randomly selected at the beginning of the exercise.

### C. Actions

The caregiver can perform only pick-and-place actions that are considered atomic by the system (i.e. they cannot be broken down to lower-level actions). The actions always result in the same effect: eliminate one token from the

```
(:action move-s3-v5
 :parameters (?from - ilocation
              ?to - glocation)
 :precondition (and
    (= ?to c4)
    (current-step s3)
    (contains ?from v5)
    (contains ?to empty)
    ; (greater-than-token-at v5 a4)
    (exists (?vaux - value)
        (and
            (contains a4 ?vaux)
            (lessthan v5 ?vaux)
        )) ; value of contents of
           ; a4 is greater than v5
    ; (greater-than-token-at v5 b4)
    (exists (?vaux - value)
        (and
            (contains b4 ?vaux)
            (lessthan v5 ?vaux)
        )) ; value of contents of
           ; b4 is greater than v5
 )
 :effect (and
    (not (contains ?from v5))
    (not (contains ?to empty))
    (contains ?to v5)
    (contains ?from empty)
 )
)
```

Fig. 5: Example of learnt action: move token *v5* at time step *s3*. The precondition requires that the destination cell is always *C4*, and that *v5* is greater than tokens at cells *A4* and *B4*. In natural language: "Token's value must be greater than the values of tokens at A4, B4. The piece must be necessarily placed at cell C4."

location where it was resting initially, and put it in another cell. Thus, the preconditions of the actions are the unknown element that INPRO seeks to learn. We consider that the rules of each exercise are determined by the actions' preconditions.

All the exercises have in common that there is a total number of 10 tokens. 5 moves are necessary to complete the exercises (a move consists in picking a token from one of the first two rows and placing it at one of the cells of the last row). INPRO aims at coming up with the model of up to 50 actions, that is, one action for each token that can be moved and for each time step in which the move could take place. The rationale is simple: in order to accommodate rules with large complexity, we allow the system to find different preconditions for the pick-and-place actions depending on the token and on the time step. Fig. 5 shows an example of an action schema in PDDL.

### D. Learning Action Preconditions via Planning

INPRO starts out with a set $A$ containing 50 action schemas that are maximally restrictive. That is, each $a \in A$, $\mathsf{pre}(a)$ contains all the possible predicates that could potentially limit the applicability of $a$. In practice, that means that no action is applicable at the beginning since their pre-

conditions contain predicates that contradict each other. For instance, if the precondition of the action shown in Fig. 5 also contained the predicate `(= ?to d4)`, the action would be inapplicable because this new predicate would be in conflict with `(= ?to c4)`. It is worth noting that preconditions may also contain existential quantifiers, as long as they are specified as input features. INPRO handles this by using proxy predicates that are later expanded to the full quantified expression. For example, `(greater-than-token-at v5 a4)` gets expanded to the first `exist` precondition of Fig. 5. Therefore, INPRO's expressiveness is limited to conjunction of input features.

INPRO progressively relaxes the action's preconditions, removing predicates until all the actions that appear in the input traces are applicable in all the states in which they were executed. Internally, INPRO operates by solving a classical planning problem, specified in PDDL. The goal is to validate all the given traces, removing preconditions if necessary. The problem defines two modes: a *programming mode*, in which predicates may be removed from the actions' preconditions; and a *validation mode*, in which the programmed actions are executed and validated across all the input traces.

This sort of problem benefits immensely from SAT planning, since parallel encoding allows exploiting that most of the actions are mostly independent of each other and can be performed in the same step. This is the case of all the operators that are employed for removing predicates from the preconditions since they do not interfere with each other. This is also true for the operators that are meant to validate actions on different traces since the execution of an action in one trace does not have any impact on the other traces. For these reasons, we use the MADAGASCAR [26] SAT planner. After each demonstration, the planner takes less than 1 second[1] to find the preconditions that should be relaxed, thus making our method suited for real time.

## V. Experimental Design

### A. Hypotheses

The main hypothesis is that interacting with a system that provides full feedback modalities will have a positive impact on user's teaching effectiveness. More formally, we aim at validating the following two hypotheses:

- **H1.** *"teachers from the minimum feedback group will require more traces to complete the exercises"*
- **H2.** *"more teachers in the full feedback group will teach all exercise rules."*.

### B. Experimental setup

We have built a board that consists of $5 \times 4$ cells, each one equipped with a 20 NFC sensor. This board constitutes the scenario in which subjects taught three different exercises to the system. The last row, called **goal row**, is where the tokens need to be arranged based on the rules of the exercise. **The storage rows** (first and second rows) are the rows where 10 tokens are initially arranged. Depending on the exercise, tokens are labelled with numbers, shapes, or letters. All the exercises require moving a total of 5 tokens from the storage rows to the goal row one at a time. The difference between exercise are provided next.

**Numbers** exercise requires users to sort 5 tokens with values less than 60 in ascending order (see Fig. 6a). Cells in the goal row must be filled from left to right without skipping any cell. For the system to learn the rules, it must be provided with at least **3 examples**.

**Letters** exercise requires users to spell out the name "CURIE" (see Fig. 6b). The users can fill the goal row either from left to right or from right to left. However, the tokens should always spell "CURIE" when reading from left to right. The system learns the rules after **2 examples**.

**Shapes** exercise is a type of a memory exercise in which the patients are provided with a sequence they need to remember and later reproduce in reverse order (see Fig. 6c). In our experiments, the participants have the role of teachers so they can see the whole sequence at all times. In this exercise, the system learns the rules after **1 example**.

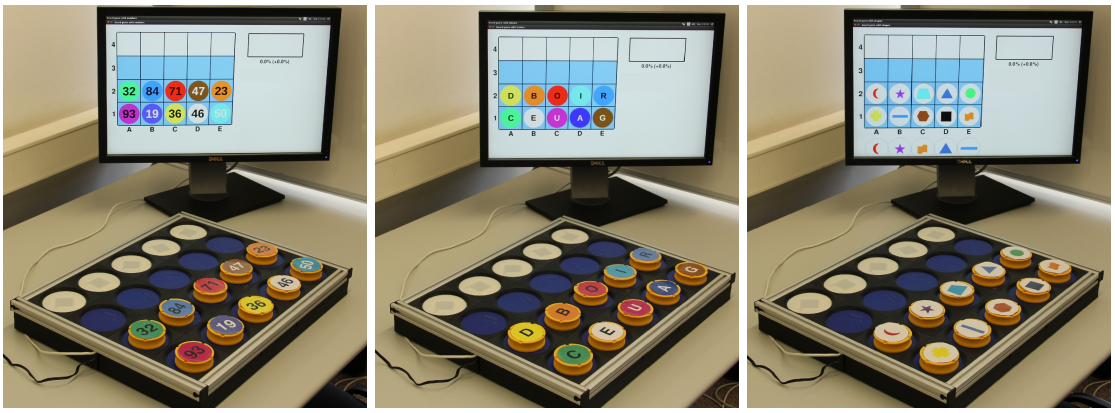
In addition to the board, an LCD screen is used to provide feedback to the participant during the exercise. An audio feedback is generated when a token is placed back at the board. The subjects were aware of the exercises' rules and, thus, we assumed that no wrong traces were provided to the system. To ensure the correctness of the traces, experimenters observed the interaction without distracting the subjects.

### C. Type of Feedback

We allocated participants into two groups: one in which the system interacts with *minimum feedback* modality; and the other where the system provides *full feedback*. The system is able to provide three types of feedback: a *learning indicator*, a *graphical representation of the learnt rules*, and a natural language description of the actions' precondition (respectively, the red, green and blue boxes shown in Fig. 7). They are described in more detail next.

The learning indicator is the percentage of relaxed preconditions in the action schemas. An increment of this indicator means that the system has learnt something from the last demonstration. If, on the contrary, its value does not increase after a demonstration, it means that the last demonstration was not informative. Notice that a 100% denotes fully relaxed rules, implying that all the preconditions have been removed and every movement from the storage rows to the goal row is valid (which is certainly not the goal). The participants were told to take into account only the increment of this indicator (shown in parenthesis) as a yes/no sign of whether the last trace was informative, and not to mind the actual percentage. INPRO may draw arrows on top of the graphical representation of the board that indicate which pick-and-place actions could be applied according to the system's current knowledge. Finally, the system provides natural language description of the learnt rules. Each feature has a template text. The template's placeholders are filled in runtime, and the resulting text is combined in a description. In the full feedback group, all three types of feedback

[1]AMD Ryzen 3700X @ 3.6GHz, 32GB RAM



(a) *Numbers* exercise (b) *Letters* exercise (c) *Shapes* exercise

Fig. 6: Initial configuration of the three exercises used as validation.

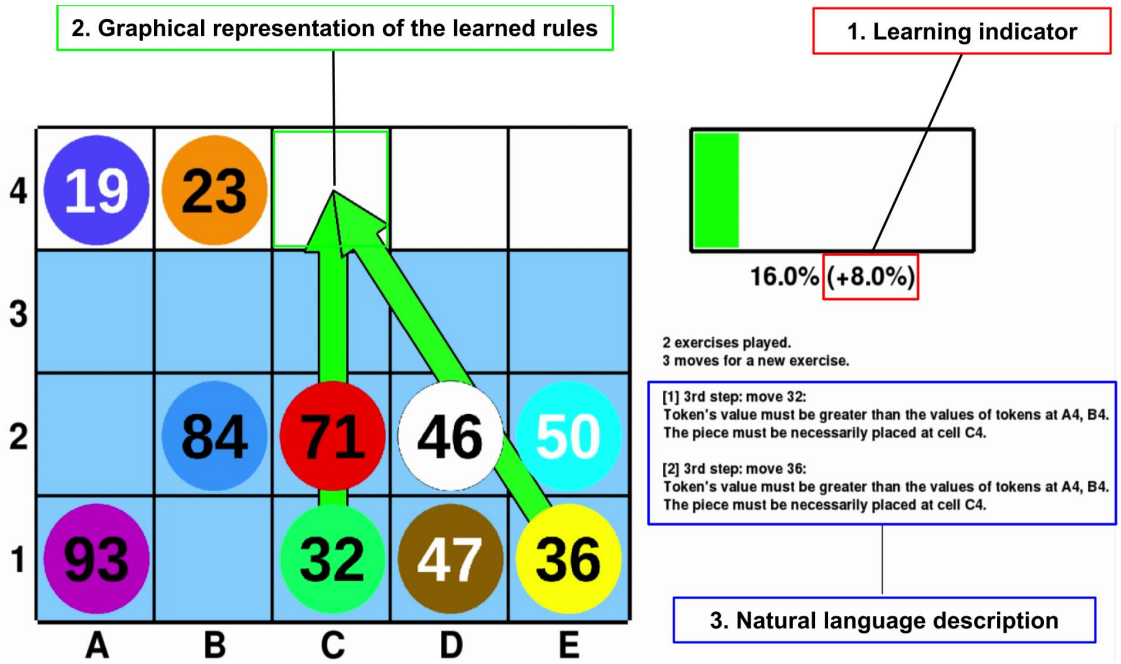



Fig. 7: Full feedback assistance

were used, while in the Minimum Feedback group only the learning indicator feedback was shown.

### D. Participants

We recruited 37 participants (9 female), aged 18-42, with no conflict of interest. Moreover, we aimed at having a diverse group of participants in terms of age and gender. All participants were randomly assigned to one of the two different feedback modalities (19 for the *full feedback* group, and 18 for the *minimum feedback* group). Participants in each group were told to teach the three exercises in different order. We uniformly distributed the participants into the 6 possible permutations of the exercises.

The goal for the participants is to provide as many examples as they deemed necessary for the system to learn the rules of each exercise. After each example, they judged whether the system learnt all the rules. If so, they had to notify the experimenter and proceed to the next exercise or to the end of the experiment. Otherwise, they kept providing examples. The participants could ask questions at any moment. However, questions related to the underlying learning algorithm were not answered until the end of the experiment, to ensure an unbiased behavior from the participants. Our paper's website[2] provides a video in which we demonstrate how the INPRO system work, as well as more information on the ethical aspects of our experiment, and the source code of INPRO.

## VI. Results and Evaluation

To evaluate the hypotheses **H1** and **H2** defined in Sec. V-A, we observed several variables: useful traces, non-informative traces, and validation traces. Useful traces are

[2]http://www.iri.upc.edu/groups/perception/#INPRO

those that contribute knowledge to the system or, in other words, that lead to relaxing some preconditions. Non-informative traces are traces that were redundant. Participants provided non-informative traces because they either were unsure that the system learnt the rules or because they wanted to confirm what it had learnt. Validation traces are much like non-informative traces, except that they are provided once the system is fully aware of the rules. They also play a confirmation role. We have extracted the average number of these metrics for each group of participants, and present the results in Table I.

The *shapes* and *letters* exercises were successfully taught by all participants with both feedback modalities (Table I). However, there is a difference in the number of traces that participants provided to teach the system. For both exercises, the participants in the *full feedback* modality group provided fewer traces, both non-informative and validation traces. The only exception is for the *shapes* exercise where the number of non-informative traces is zero for both feedback modalities because one trace suffices to teach the system all the rules. While *shapes* and *letters* are not complex, they are examples of a broad class of cognitively interesting games with simple rules that can be learnt by INPRO.

Taken together, these results suggest that teachers belonging to the group with *minimum feedback* needed in average more traces to teach the system the correct exercises. This is aligned to our hypothesis **H1**. As the number of minimum traces for both exercises is small, and the rules are simple, all the participants, regardless of the groups they belong, were able to fully teach the rules. Thus, hypothesis **H2** is not tested yet.

The *numbers* exercise was more challenging for the participants because it has multiple solutions, and some participants did not manage to teach the whole set of rules. To validate hypothesis **H1**, we analyzed only the participants that were able to finish the *numbers* exercise (12 participants, 32% of the total number of participants). As in the case of *shapes* and *letters*, participants that were provided with full feedback were able to teach the exercise with fewer validation traces and total traces (see Table I). Thus our initial hypothesis **H1** is also valid in the *numbers* exercise.

Additionally, we can evaluate the errors committed by the subjects. To do so, we identify the ground truth of deleted predicates as $P$ (i.e. the predicates that have to be necessarily removed to achieve the desired rules). On the other hand, the predicates that were removed by INPRO as per a set of given traces are called $P'$. Since we assume that traces are always correct, $P' \subseteq P$. We define the accuracy in terms of the *Jaccard index* between these two sets:

$$\mathrm{Acc}(P') = \mathrm{J}(P', P) = \frac{|P' \cap P|}{|P' \cup P|} \tag{1}$$

Thus, the error is defined as $\mathrm{E}(P') = 1 - \mathrm{J}(P', P)$. Figure 8 shows, for each error value between 0 and 0.40, the percentage of subjects from both groups that have achieved the same or less error. Some of the participants in both groups failed to teach the system all the rules. However, more participants in the *full feedback* modality group (42% of the group) managed to teach all the relevant rules compared to *minimum feedback* modality group (22% of the group). It is noteworthy that for error smaller than 0.08, the proportion of participants from *full feedback* modality is considerably bigger in comparison with the group of *minimum feedback*. In summary, participants from the group with *minimum feedback* struggled more on teaching all the correct rules. Hence, the obtained results support our initial hypothesis **H2**.

TABLE I: User performance while teaching the rules for three exercises, with two different feedback modalities.

| | **Shapes** | | **Letters** | | **Numbers** | |
|---|---|---|---|---|---|---|
| | **Min.** | **Full** | **Min.** | **Full** | **Min.** | **Full** |
| **useful traces** | 1.00 | 1.00 | 2.00 | 2.00 | 3.00 | 4.00 |
| **noninformative traces** | 0.00 | 0.00 | 0.33 | 0.21 | 0.25 | 0.38 |
| **validation traces** | 2.44 | 1.63 | 1.72 | 1.58 | 4.00 | 1.62 |
| **total traces** | 3.44 | 2.63 | 4.05 | 3.79 | 7.25 | 6.00 |

Despite that, the trend is not so clear for errors higher than 0.08, i.e. when we involve the users who perform worse in the experiment. We believe this is caused by the apparently large number of combinations in the *numbers* exercise. While the system can infer all the rules from a minimum of 3 traces, some participants (independently of the group) may have felt compelled to try out every possible solution (there are 7 tokens smaller than 60, out of which only 5 can be selected, therefore $\binom{7}{5} = 21$ possible solutions), losing track and failing to give truly informative traces. We think that the graphical and textual feedback help with this issue and that is why at the end ($error < 0.08$) we get more successful results in the group with full feedback.

We conducted an informal poll among the participants from the group with *full feedback*. The answers suggest that the most helpful aid is the graphical representation of the applicable actions in the form of arrows. However, when the graphical representation was ambiguous (i.e. overlapping arrows), participants would rely on the natural language description.

## VII. Conclusions and Future Work

In this work, we have explored a *system-centric* approach for learning action preconditions. This approach is based on classical planning and SAT-based solvers [5]. For future lines of research, we will explore other promising research directions to increase the complexity of the learned rules, like *object-centric* approaches [27] and *unification-based* ones [19]. Furthermore, we would like to consider allowing the specification of forbidden moves (negative examples).

We performed a user-study involving 37 participants. In our proof-of-concept experiments, we analyzed qualitatively whether including autonomous feedback sped up the human-robot teaching process, resulting in a positive impact for the human teacher. As future work, we would like to test the framework including different autonomous feedback options and harder problems with a large group of people (preferably professional caregivers, who are the target audience of this

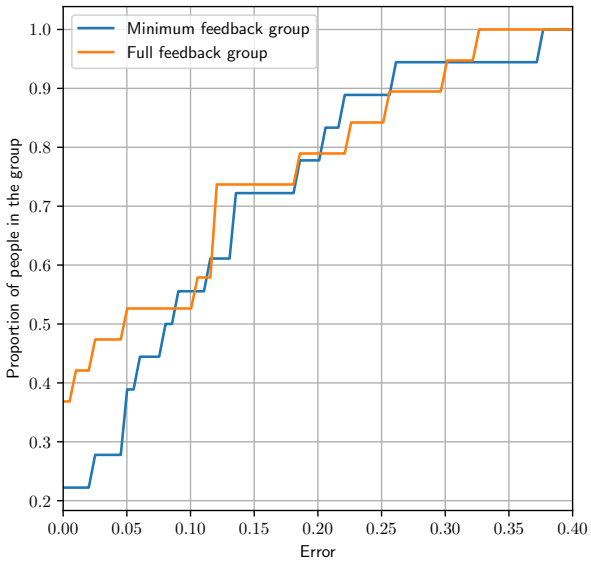



Fig. 8: Cumulative proportion of teaching errors. Observe between error 0 and 0.08 that the full feedback group managed to teach better the new exercise.

work), so that our hypotheses can be statistically accepted or rejected.

Finally, we would like to enable the robot to communicate through voice, gestures, and facial expressions with the intention of improving the communication of the learned rules. These same features are useful when monitoring patients [28], so we suspect they can also have a positive impact during the teaching process.

## REFERENCES


[1] D. Feil-Seifer and M. J. Mataric, "Defining socially assistive robotics," in *9th International Conference on Rehabilitation Robotics, 2005. ICORR 2005.*, June 2005, pp. 465–468.

[2] W. Moyle, M. L. Cooke, E. Beattie, C. Jones, B. Klein, G. Cook, and C. Gray, "Exploring the effect of companion robots on emotional expression in older people with dementia : a pilot randomized controlled trial," *Journal of Gerontological Nursing*, vol. 39, no. 5, pp. 46–53, 2012.

[3] I. Tiddi and E. al., "A user-friendly interface to control ROS robotic platforms," *CEUR Workshop Proc.*, vol. 2180, 2018.

[4] I. Zubrycki, M. Kolesiński, and G. Granosik, "Graphical programming interface for enabling non-technical professionals to program robots and internet-of-things devices," *Adv. Comput. Intell.*, vol. 10306, pp. 620–631, 2017.

[5] D. Aineto, S. J. Celorrio, and E. Onaindia, "Learning action models with minimal observability," *Artificial Intelligence*, vol. 275, pp. 104–137, 2019.

[6] D. Lotinac, J. Segovia-Aguas, S. Jiménez, and A. Jonsson, "Automatic generation of high-level state features for generalized planning," in *Int. Joint Conference on Artificial Intelligence*, 2016, pp. 3199–3205.

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

[8] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 1.

[9] J. E. Overall and R. Schaltenbrand, "The skt neuropsychological test battery," *Topics in geriatrics*, vol. 5, no. 4, pp. 220–227, 1992.

[10] C. de Boer, H. V. Echlin, A. Rogojin, B. R. Baltaretu, and L. E. Sergio, "Thinking-while-moving exercises may improve cognition in elderly with mild cognitive deficits: a proof-of-principle study," *Dementia and geriatric cognitive disorders extra*, vol. 8, pp. 248–258, 2018.

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

[12] D. Martínez, G. Alenyà, T. Ribeiro, K. Inoue, and C. Torras, "Relational reinforcement learning for planning with exogenous effects," *Journal of Machine Learning Research*, vol. 18, no. 78, pp. 1–44, 2017.

[13] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *Journal of Machine Learning Research*, vol. 10, no. Jul, pp. 1633–1685, 2009.

[14] Y. Gil, "Learning by experimentation: Incremental refinement of incomplete planning domains," in *Machine Learning Proceedings 1994*. Elsevier, 1994, pp. 87–95.

[15] S. Benson, "Inductive learning of reactive action models," in *Machine Learning Proceedings 1995*. Elsevier, 1995, pp. 47–54.

[16] X. Wang, "Learning by observation and practice: An incremental approach for planning operator acquisition," in *Machine Learning Proceedings 1995*. Elsevier, 1995, pp. 549–557.

[17] 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.

[18] E. Amir and A. Chang, "Learning partially observable deterministic action models," *Journal of Artificial Intelligence Research*, vol. 33, pp. 349–402, 2008.

[19] 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, May 2021. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/17423

[20] E. Senft, S. Lemaignan, M. Bartlett, P. Baxter, T. Belpaeme *et al.*, "Robots in the classroom: Learning to be a good tutor," in *Robotrs for Learning (R4L) workshop at HRI2018*, 2018.

[21] 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," *Proceedings of Robotics: Science and Systems. Corvalis, Oregon, USA. https://doi. org/10.15607/RSS*, 2020.

[22] H. Geffner and B. Bonet, "A concise introduction to models and methods for automated planning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 8, no. 1, pp. 1–141, 2013.

[23] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "Pddl-the planning domain definition language," 1998.

[24] R. E. Fikes and N. J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," *Artificial intelligence*, vol. 2, no. 3-4, pp. 189–208, 1971.

[25] A. Arora, H. Fiorino, D. Pellier, M. Métivier, and S. Pesty, "A review of learning planning action models," *The Knowledge Engineering Review*, vol. 33, 2018.

[26] J. Rintanen, "Madagascar: Scalable planning with sat," *Proceedings of the 8th International Planning Competition (IPC-2014)*, vol. 21, 2014.

[27] S. Cresswell, T. L. McCluskey, and M. M. West, "Acquisition of object-centred domain models from planning examples." in *ICAPS*, 2009.

[28] A. Andriella, H. Siqueira, D. Fu, S. Magg, P. Barros, S. Wermter, C. Torras, and G. Alenya, "Do i have a personality? endowing care robots with context-dependent personality traits," *International Journal of Social Robotics*, 2020.