Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 176 (2020) 440–449

Procedia Computer Science

www.elsevier.com/locate/procedia

24th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

Towards inductive learning of surgical task knowledge: a preliminary case study of the peg transfer task☆

Daniele Meli[a,*], Paolo Fiorini[a], Mohan Sridharan[b]

[a]*Department of Computer Science, University of Verona, strada Le Grazie 15, Verona, 37135, Italy*
[b]*School of Computer Science, University of Birmingham, Edgbaston, Birmingham, B152TT, UK*

Abstract

Autonomy in robotic surgery will significantly improve the quality of interventions in terms of safety and recovery time for the patient, and reduce fatigue of surgeons and hospital costs. A key requirement for such autonomy is the ability of the surgical system to encode and reason with commonsense task knowledge, and to adapt to variations introduced by the surgical scenarios and the individual patients. However, it is difficult to encode all the variability in surgical scenarios and in the anatomy of individual patients a priori, and new knowledge often needs to be acquired and merged with the existing knowledge. At the same time, it is not possible to provide a large number of labeled training examples in the robotic surgery. This paper presents a framework based on inductive logic programming and answer set semantics for incrementally learning domain knowledge from a limited number of executions of basic surgical tasks. As an illustrative example, we focus on the peg transfer task, and learn state constraints and the preconditions of actions starting from different levels of prior knowledge. We do so using a small dataset comprising human and robotic executions with the da Vinci surgical robot in a challenging simulated scenario.

Keywords: Incremental knowledge acquisition; Surgical robotics; Inductive logic programming; Answer set programming

1. Introduction

In the last few decades, the use of robots in the operating room has helped surgeons perform minimally invasive surgery, improving the precision of surgeons and the recovery time of patients [7, 27, 40]. At present, surgeons use a master console to tele-operate slave manipulators acting on the patient. One long-term goal of research in surgical robotics is the development of cognitive robot systems capable of executing a surgical operation, or at least a part

☆ This research has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, ARS (Autonomous Robotic Surgery) project, grant agreement No. 742671.

* Corresponding author.
E-mail address: daniele.meli@univr.it

1877-0509
10.1016/j.procs.2020.08.046

of it, with minimal supervision of a human expert [4, 30], which can boost safety, optimize resource usage, and reduce patient recovery time, surgeon fatigue, and hospital costs [41]. The complexity of surgical scenarios makes it difficult to encode comprehensive domain knowledge or provide many labeled training examples. Hence, autonomy requires the robot to reason with incomplete commonsense domain knowledge, and adapt automatically to variations in the surgical scenario and individual patients. Our recent framework integrated logic-based reasoning about task-level actions with adaptive motion planning and control, for the automated execution of a surgical training task, the peg transfer [18]. In this framework, task-level reasoning was based on Answer Set Programming (ASP), a non-monotonic logic programming paradigm [16]. Logic programming helps to elegantly encode high-level specifications extracted from expert knowledge, and constraints on the behavior of robots, especially in the context of safe and reliable operation in dynamic domains. Moreover, ASP supports non-monotonic logical reasoning, i.e., the retraction of previously held beliefs, which is an important ability in robotics applications. A key limitation of our prior work was that it assumed comprehensive knowledge of the domain and the tasks in terms of domain attributes (e.g., object properties) and axioms governing domain dynamics (e.g., constraints, and action preconditions and effects) [18]. This is not a realistic requirement in practical robotics domains, especially in surgical scenarios that are characterized by a large number of variations in the patient's anatomy that are not known in advance and may change over time.

In this paper, we focus on the problem of learning previously unknown knowledge from a small number of examples of a surgical training task, the peg transfer task. We describe a framework that uses the Inductive Logic Programming (ILP) formalism and labeled example executions of the task to learn axioms governing domain dynamics, with an underlying system based on answer set semantics for representing and reasoning with incomplete domain knowledge. We demonstrate the capabilities of this framework in two phases. In the first phase, initial incomplete knowledge about the task and the domain is learned from limited human executions of the task in simple scenarios. We show that it is possible to refine this knowledge for execution in more complex scenarios adding examples that we obtained from the surgical da Vinci robot for our previous work [18]. In the second phase, we assume that no human demonstrations are available and use the examples obtained from the da Vinci robot to learn task specifications, given only the definition of the domain attributes. Experimental results indicate that it is possible to acquire domain knowledge from a limited set of examples. This is particularly advantageous in surgery, where standard black-box or probabilistic machine learning techniques do not perform well due to the limited availability of labeled examples. In addition, the underlying logic-based distributed representation of knowledge helps make the reasoning system more transparent, promoting ease of development, reliability, and safety. Such learning and automation is novel in the surgical robotics scenario.

We first motivate the proposed framework by reviewing related work in Section 2. Section 3 describes the peg transfer task, its formulation in ASP, the ILP-based formulation of the task of learning knowledge of domain dynamics, and the tool we use for learning. Section 4 describes the experimental set up, and compares the learned specifications with the original description in our prior work [18], with and without prior knowledge. Finally, Section 5 summarizes the key contributions and directions for future research.

2. Related Work

Building a surgical process model (SPM) requires us to choose the level of granularity at which the task will be analyzed; we use the definition of granularity (for surgical processes) described in [22] and shown in Figure 1. Learning an SPM for motions and activities, i.e., motions associated with a specific semantic meaning, surgical instrument, and hand, is a challenging task since surgical gestures present high variability (e.g., the suturing action differs significantly depending on the shape of the wound, the needle, the type of tissue) [32]. Statistical methods such as Gaussian mixture models (GMMs) [26, 35] and hidden Markov models (HMMs) [39, 42] are typically used to capture this variability and infer a low-level SPM [22]. With the increasing use of deep neural networks, GMMs have been combined with deep reinforcement learning to transfer learned motions to different surgical tools [6]. Although deep learning provides a small improvement in accuracy over traditional learning methods, it is computationally expensive to tune the network parameters [12, 14]. A further limitation of deep networks and related methods is that they need many (labeled) examples to build a good SPM. We recently proposed a framework based on dynamic movement primitives to learn gestures from a limited set of examples [18], but scaling to more complex surgical tasks is an open problem.

In this paper, we focus on learning SPMs at a coarser granularity, i.e. at the level of relations between activities and steps that constitute a surgical phase. This sequence of actions for any surgical procedure is affected by the

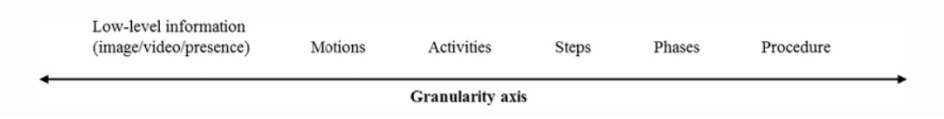


Fig. 1: Standard granularity levels of surgical processes, as described in [22].

variations in the anatomical conditions. Statistical methods such as Bayesian networks (BNs) represent the state of the art for learning SPMs at this granularity [3, 5]. Recurrent (deep) neural networks have also been explored, providing similar results evidenced at finer granularity [10]. However, most of the existing work (in surgical robotics) focuses on recognizing the surgical steps being performed by a surgeon in order to provide useful support. For instance, an HMM is used to model a peg transfer (surgical) training exercise involving cooperation between a human and a robot [2]. Only partial automation is supported in this work, with manual control required for critical interactions. Also, although they consider a simplified version of this exercise, 80 labeled human executions are required as a training set for learning, making scalability to more complex tasks a challenging problem. Another major limitation of data-driven (especially the "end to end") methods is that they generate black-box models that do not provide any guarantees in terms of correctness and soundness, affecting the reliability of the surgical system. Logic-based formalisms for representing and reasoning with domain knowledge inherently provide correctness guarantees [32], and they make the underlying reasoning easier to understand, which is an appealing capability for surgical robots. However, such logic-based formalisms for the peg transfer task have required comprehensive domain knowledge to be encoded a priori [18, 19], which is rather difficult to do in more complex surgical scenarios.

There is an established literature in AI on methods for learning domain knowledge. Examples include the incremental revision of a first-order logic representation of action operators [17], the expansion of the theory of actions to revise or inductively learn ASP system descriptions [1, 25], and the coupling of non-monotonic logical reasoning, inductive learning, and relational reinforcement learning to incrementally acquire previously unknown actions and their preconditions and effects [38]. Previously unknown state constraints have also been learned using decision tree induction in a framework that combines ASP-based non-monotonic logical reasoning with deep learning for scene understanding [29]. All these approaches may be viewed as instances of interactive task learning, a general framework for acquiring domain knowledge using labeled examples or reinforcement signals obtained from domain observations, demonstrations, or human instructions [21].

Our framework uses ILP to learn previously unknown domain axioms represented as ASP programs. First developed in [31], ILP allows to learn from typically few labeled examples. Hence it has been used by an international research community in different domains, e.g., to identify cognitive stress and distraction in a driver [28]; for event recognition in city transport [20]; and to learn logic programs in robotics [8]. ILP has been used to learn in Prolog programs [34], non-monotonic logic programs [25] and probabilistic logic programs [9]. In complex application domains such as surgical robotics, learning with probabilistic logics is computationally challenging [33], but non-monotonic logical reasoning is still necessary. We thus chose to build on ILASP, an implementation of ILP under the AS semantics [23]. Unlike another such implementation called Inspire[37], ILASP has fewer hyper-parameters and supports learning from a batch of examples, which significantly speeds up the learning process.

3. Materials and methods

In this section, we first describe "peg transfer", our running example of a surgical training task (Section 3.1. Section 3.2 describes this task's encoding in ASP, as introduced in prior work [18], and Section 3.3 provides the ILP formulation for learning previously unknown axioms under the AS semantics.

3.1. The peg transfer task

Figure 2 shows the setup for the peg transfer task; the objective is to place colored rings on pegs of the corresponding color using the two slave manipulators (PSM1, PSM2) of the da Vinci surgical robot. The standard definition of this tasks requires all rings to be transferred between the two arms before being placed on the pegs [11]. We consider

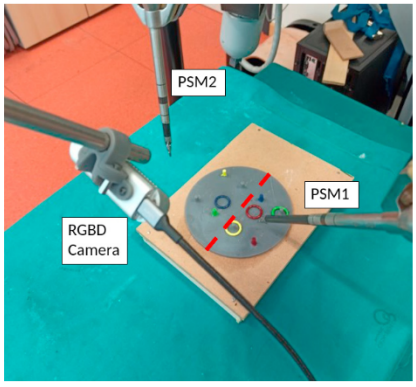


Fig. 2: Setup for the peg transfer task. The red dashed line marks the reachability regions for the two arms of the da Vinci robot, PSM1 and PSM2.

a variant of this definition in which an arm can grasp any reachable ring and place it on any reachable peg; reachability is determined by the relative position of rings and pegs with respect to the center of the base. We also consider anomalies and complexities not included in the standard definition, e.g., pegs can be occupied by other rings and must be freed before placing another ring on it. Also, rings may be on pegs or on the base in the initial state. The spatial information necessary for task completion, e.g., relative position of rings and pegs and with respect to the base, are retrieved from a RGBD camera.

3.2. ASP encoding of peg transfer task

ASP is a declarative language that can represent recursive definitions, defaults, causal relations, and constructs that are difficult to express in classical logic formalisms [16]. It encodes concepts such as *default negation* (negation by failure) and *epistemic disjunction*, e.g., unlike "¬a", which implies that "*a is believed to be false*", "*not a*" only implies "*a is not believed to be true*", i.e., each literal can be true, false or unknown. ASP supports non-monotonic logical reasoning, i.e., adding a statement can reduce the set of inferred consequences. Modern ASP solvers support efficient reasoning with large knowledge bases or incomplete knowledge, and are used by an international community [13].

A domain's description in ASP comprises a *system description* $\mathscr{D}$ and a *history* $\mathscr{H}$. $\mathscr{D}$ comprises a *sorted signature* Σ and axioms. Σ includes *basic sorts* arranged hierarchically; *statics*, i.e., domain attributes that do not change over time; *fluents*, i.e., domain attributes whose values can be changed, and *actions*; domain attributes and actions are defined by the sorts of their arguments. Variables and object constants are *terms*; terms with no symbols or variables are *ground*. A predicate of terms is an *atom*; it is *ground* if all its terms are ground. An atom or its negation is a *literal*. For the peg transfer task, sorts include location (including specific instance `center`) and `step` for temporal reasoning; statics include `object` (with subsorts `ring` and `peg`), the `color` of a ring or peg (red, green, blue, yellow, grey), and the robot's `arm` (PSM1 and PSM2); and fluents include `reachable(arm, object, color)`, `in_hand(arm, ring, color)`, `on(ring, color, peg, color)`, `at(arm, center)`, `closed_gripper(arm)`, and `at(arm, object, color)`. Actions include `move(arm, object, color)`, `move(arm, center)`, `grasp(arm, ring, color)`, `extract(arm, ring, color)` and `release(arm, ring, color)`. Given this Σ, axioms describing domain dynamics are specified as statements in an action language, e.g.:

$$move(A,O,C) \ \textbf{causes} \ at(A,O,C) \tag{1a}$$

$$grasp(A,R,C) \ \textbf{causes} \ in\_hand(A,R,C) \tag{1b}$$

$$\neg on(R,C_1,P,C) \ \textbf{if} \ on(R,C_2,P,C) \tag{1c}$$

$$\textbf{impossible} \ move(A,P,C) \ \textbf{if} \ on(R,\_,P,C) \tag{1d}$$

$$\textbf{impossible} \ move(A,P,\_) \ \textbf{if} \ in\_hand(A,R,C), \ on(R,C,P,\_) \tag{1e}$$

where "_" implies that the instance of the corresponding sort is irrelevant to the axiom, and capital letters are variables. Statements 1(a-b) are causal laws encoding the outcomes of actions `move` and `grasp`; (c) is a state constraint preventing two rings from being on the same peg; and (d-e) are executability conditions; (d) forbids the move of an arm to a peg

that already has a ring, and (e) prevents the move of a ring placed on a peg (it requires extraction first). History $\mathscr{H}$ is then a record of observations of values of fluents, and of the execution of actions, at particular time steps.

To reason with domain knowledge, we construct the ASP program $\Pi(\mathscr{D}, \mathscr{H})$ that includes the signature and axioms of $\mathscr{D}$, inertia axioms, reality checks, closed world assumptions, observations and actions from $\mathscr{H}$, and the goal (if appropriate). For instance, Statements 1(a-e) are translated to:

```
at(A, O, C, t+1):- move(A, O, C, t)                                   (2a)
in_hand(A, R, C, t+1) :-  grasp(A, R, C, t)                           (2b)
:- on(R, C1, P, C, t), on(R, C2, P, C, t), C1 != C2                   (2c)
:- move(A, P, C, t), on(R, _, P, C, t)                                (2d)
:- move(A, P, _, t), in_hand(A, R, C, t), on(R, C, P, _, t)           (2e)
```

where `:-` $\equiv \leftarrow$, Statements 2(c-e) imply that literals on the right of `:-` cannot hold concurrently, and "t" refers to a discrete time step. Adding "t" as an additional argument is short hand for stating that the corresponding fluent (action) holds (occurs) at a particular time step, e.g., `at(A, O, C, t)` and `grasp(A, R, C, t)` instead of `holds(at(A, O, C), t)` and `occurs(grasp(A, R, C), t)` respectively. The goal for our task is to have all rings on pegs of matching color:

```
:- reachable(_, R, C), reachable(_, P, C), not on(R, C, P, C, t)
```

Planning, diagnostics, and inference can then be reduced to computing *answer sets* of Π; an answer set describes a possible world in terms of the beliefs of an agent associated with Π. We compute answer sets using Clingo [15]. For the peg transfer task, we are primarily interested in atoms of action, and relevant fluents and statics; for simplicity, we will focus on only these atoms in our description of answer sets below. Also, we only focus on learning the pre-conditions of actions and the executability conditions. To facilitate this, actions will be rewritten as:

```
0 {move(A, O, C, t) : reachable(A, O, C, t); move(A, center, t) : in_hand(A, R, C, t);   (3)
extract(A, R, C, t) : in_hand(A, R, C, t); grasp(A, R, C, t) : at(A, R, C, t);
release(A, R, C, t) : in_hand(A, R, C, t) } 1
```

where `0 { Action: Pre-condition } 1` defines a constraint on the number of elements which can be selected from a set of atoms, i.e., only one of these actions can be executed at a time. Furthermore, we will focus on learning Statements 2(d-e) and the following three executability conditions:

```
:- move(A1, P, _, t), in_hand(A1, R, C, t), in_hand(A2, R, C, t), A1 != A2   (4a)
:- move(A, center, t), in_hand(A, R, C, t), on(R, C, P, _, t)                (4b)
:- move(A, R, _, t), closed_gripper(A, t)                                    (4c)
```

where Statement 4(a) guarantees that neither arm moves if they are both holding the same ring (during transfer); Statement 4(b), similar to Statement 2(e), prevents the move of a ring which is placed on a peg; and Statement 4(c) forbids movement of an arm to a ring if the gripper is closed.

3.3. Inductive learning from answer sets and ILASP

Next, we describe the ILP formulation for the learning of axioms under AS semantics. To do so, we adapt the related definitions from [25] to surgical training tasks.

Definition 1 (Induction task under AS semantics)**.** Let B be the *background knowledge* for the task, S_M be the *search space*, i.e. a set of logic formulas called *hypotheses*, and let B and S_M be expressed in the AS syntax; in particular, let S_M be a set of rules, i.e. logical implications. Let E be the *examples*, i.e. the answer sets to learn from. The *learning task* for ILP is defined as: given $\mathcal{T} = \langle B, S_M, E \rangle$, find a set $H \subseteq S_M$ such that $H \cup B \models E$.

In our case, background knowledge (B) comprises the known domain literals and axioms in the peg transfer task, the search space (S_M) is the set of potential axioms we can hypothesize, and examples (E) include the actual answer sets obtained (by the robot) as experiences. The objective of the ILP learning task is to find the (minimal) set of the possible axioms to accept (H) to *explain* the examples. Also, the set of examples may be split as $E = \langle E^+, E^- \rangle$ into

the set of *positive examples* (E^+) and the set of *negative examples* (E^-). Intuitively, positive examples are answer sets which are expected to be entailed by $H \cup B$, while negative examples are to be excluded.

Next, we define two sub-tasks of Definition 1, *brave induction* and *cautious induction* as presented in [36].

Definition 2 (Brave induction task)**.** Let $\mathcal{T} = \langle B, S_M, E \rangle$, H, and $E = \langle E^+, E^- \rangle$ be as defined above. Let A be an answer set of the ASP program defined by $H \cup B$. $\mathcal{T}$ is said to define a *brave induction task* if the goal set H of hypotheses must satisfy:

$$\exists A \; s.t. \; B \cup H \models A : \; A \cap E^- = \emptyset \wedge E^+ \subseteq A$$

Definition 3 (Cautious induction task)**.** Let $\mathcal{T} = \langle B, S_M, E \rangle$, H, and $E = \langle E^+, E^- \rangle$ be as defined above. Let A be an answer set of ASP program defined by $H \cup B$. $\mathcal{T}$ is said to define a *cautious induction task* if the goal set H of hypotheses must satisfy:

$$\forall A \; s.t. \; B \cup H \models A : \; A \cap E^- = \emptyset \wedge E^+ \subseteq A$$

In other words, the hypotheses H selected by a brave induction task must entail *at least* one answer set that satisfies the examples, and the H returned by a cautious induction task must entail *only* answer sets that satisfy the examples. For any set of examples, hypotheses returned by a cautious induction task are stricter than those returned by a brave induction task since it must satisfy additional constraints.

In this paper, we use the tool ILASP [23] to learn inductively from answer sets. ILASP combines the definitions of brave and cautious induction, defining a new learning task as follows.

Definition 4 (ILASP task)**.** Let $\mathcal{T} = \langle B, S_M, E \rangle$, H, and $E = \langle E^+, E^- \rangle$ be defined as before. Let A be an answer set from the ASP program defined by $H \cup B$. The goal of the ILASP task is to find H such that:

$$\forall e \in E^+ \; \exists A \; s.t. \; B \cup H \models A : \; e \subseteq A$$
$$\forall e \in E^- \; \nexists A \; s.t. \; B \cup H \models A : \; e \subseteq A$$

The hypotheses returned by ILASP hence *bravely entail positive examples* and help learn preconditions, and *cautiously entail negative examples* and help learn executability conditions.

ILASP allows us to define the search space S_M with compact syntax, using *mode bias* that specifies the atoms that can occur in the body and head of rules. For the peg transfer task, we allow only atoms identifying actions to be part of the heads of candidate rules; the body of rules can have both actions and domain attributes. Since the learning time depends on the size of the search space, we first learn specifications for single actions, running parallel ILASP tasks for each of them. Then, we run a complete ILASP task for all actions, adding the learned specifications in the background knowledge to validate them.

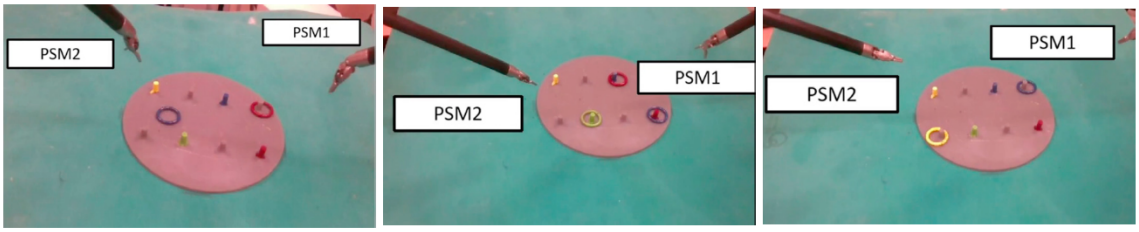
Another feature of ILASP is that it is designed to find the *minimal* H in the search space S_M. To explain this feature, we first define the *length* of a rule.

Definition 5 (Length of a rule)**.** Let $\mathcal{R}$ be an axiom in an ASP program. The *length* of $\mathcal{R}$, $|\mathcal{R}|$, is defined as the number of atoms that appear in it. For an aggregate rule, i.e. a rule with an aggregate $l \; \{a_1; a_2; \ldots; a_n\} \; u$ in the head, the length of the head is defined as $\sum_{i=l}^{u} i \cdot n$.

The minimal set H is then the set of rules in S_M with minimal length that satisfy the goal of ILASP task. We thus build on our ASP-based representation and reasoning system to formulate the task of discovering previously unknown preconditions and executability conditions as an ILP problem. This coupling of reasoning and learning allows us to apply ILASP to search for the minimal set of new axioms that explain the examples (i.e., observed answer sets).

4. Experimental results

Recall that in a departure from existing work, our work builds on a logic-based formalism but focuses on incrementally learning the domain axioms. This section describes the dataset used for the preliminary case study of the peg transfer surgical training task, and the results obtained with varying amounts of knowledge encoded a priori.



(a) (b) (c)

Fig. 3: Screenshots of unconventional initial states of surgical robot executing action sequences that form the dataset for learning.

4.1. Labeled examples

For the peg transfer task, we use videos of human and autonomous robot executions of the task, which we collected recently [18]. The data includes 15 human executions that are used to learn motion primitives. Each such demonstration involves a human tele-operating the slave manipulators using the master console. In each execution, all four rings are in the scene on grey pegs, and they must be transferred between the two arms before being placed on suitable colored pegs. Hence, all actions mentioned in Section 3 appear in the videos. The videos differ only in the sequence of rings manipulated during the procedure; since all executions provide the same semantic content, we only annotate one execution. In addition, screenshots of initial states of some automated executions by the robot are shown in Figure 3. These images show unconventional conditions and failures, useful to learn the task knowledge. Specifically, in Figure 3a the transfer of the blue ring fails, and PSM2 must re-open the gripper before moving to the blue ring again. This experience serves as an example that can be used to learn the constraint in Statement 4(c) above. Furthermore, the blue ring needs no extraction, differently from the human execution. In Figure 3b, blue and red pegs are occupied, hence one must be freed to complete the task. This allows to learn constraint encoded by Statement 2(d). Finally, Figure 3c refers to the parallel execution of the two arms. This is needed to guarantee that arms can operate independently without forcing sequential execution in the rules.

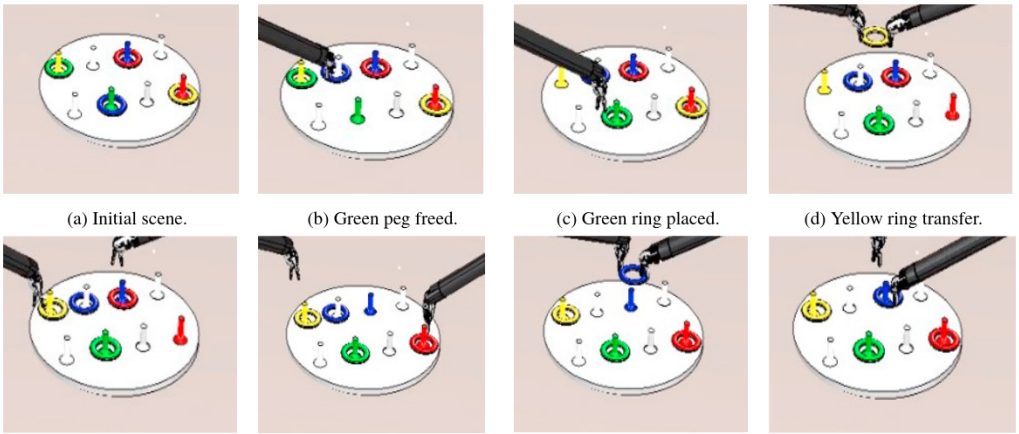
The video of each execution is manually annotated with atoms defining actions and domain attributes (i.e., fluents and statics) for the peg transfer task. Specifically, each frame in which an action begins is annotated with the name of the action and the domain attributes that hold in that state. These state variables describe geometric properties of the scene (e.g., distances between objects) and can be retrieved for each frame. Actions occurring in each frame define the positive examples for the ILASP task. To learn executability constraints, we also label each frame with the actions that cannot occur, given the knowledge of state variables that hold true, generating the negative examples. The positive and negative examples define the actual (or forbidden) action and the set of holding state variables, exploiting learning from context-dependent examples, as described in [24]. For instance, for the scene depicted in Figure 3a, the first action moves PSM1 towards the red ring, providing a positive example:

```
#pos{ex1, {move(psm1,ring,red)},
    {reachable(psm1,ring,red), reachable(psm2,ring,blue), reachable(psm1,peg,red),
    reachable(psm1,peg,blue), reachable(psm2,peg,green), reachable(psm2,peg,yellow),
    reachable(psm1,peg,grey), reachable(psm2,peg,grey), on(ring,red,peg,grey)}}
```

At the same time, moving PSM1 to the blue ring is not possible, providing the negative example:

```
#neg{ex2, {move(psm1,ring,blue)},
    {reachable(psm1,ring,red), reachable(psm2,ring,blue), reachable(psm1,peg,red),
    reachable(psm1,peg,blue), reachable(psm2,peg,green), reachable(psm2,peg,yellow),
    reachable(psm1,peg,grey), reachable(psm2,peg,grey), on(ring,red,peg,grey)}}
```

Similarly, we build the negative example to specify that PSM2 cannot move to the red ring. Since each example refers to a specific time step during task execution, we do not specify the time parameter in the atoms but add it later for validation. Moreover, we omit redundant negative examples for simplicity. Overall, we obtain 24 positive and 15 negative examples from human demonstration; 16 positive and 7 negative examples from execution in Figure 3a; 15 positive and 5 negative examples from execution in Figure 3b; 9 positive examples from execution in Figure 3c.



(a) Initial scene. (b) Green peg freed. (c) Green ring placed. (d) Yellow ring transfer.

(e) Yellow ring placed. (f) Red ring placed. (g) Blue ring transfer. (h) Blue ring placed.

Fig. 4: Main steps during a particular execution example observed during validation are highlighted.

4.2. Testing and validation

The robot first learns from examples of human execution, with prior knowledge including Σ, statics, initial conditions, and axioms other than the preconditions and executability conditions being learned. This results in the learning of the following action preconditions:

```
0 {move(A, O, C, t) : reachable(A, O, C, t); move(A, center, t) : in_hand(A, R, C, t);   (7)
extract(A, R, C, t) : in_hand(A, R, C, t); grasp(A, R, C, t) : at(A, R, C, t);
release(A, R, C, t) : in_hand(A, R, C, t) } 1
```

and the following executability conditions:

```
:- move(A, P, _, t), in_hand(A, R, C, t), on(R, C, P, _, t)                    (8a)
:- move(A1, P, _, t), at(A2, center, t), closed_gripper(A2, t)                 (8b)
:- move(_, center, t), in_hand(A, R, C, t), on(R, C, P, _, t)                  (8c)
```

Statement 7 matches the action preconditions in Statement 3. Statements 8(a-c) represent the same conditions as in Statements 2(e), 4(a) (forbidding motion when both arms hold the same ring is equivalent to forbidding motion when the other arm is at center and holding the ring, after transfer) and 4(b) respectively. However, the robot's knowledge is incomplete since the colored pegs are always free in the human execution examples and no failure conditions are demonstrated. These axioms are inserted in the ASP program and the learning task is repeated with the examples of the robot's execution of the peg transfer task. This results in the following additional axioms being learned:

```
:- move(A, R, _, t), closed_gripper(A, t).                                     (9a)
:- on(R, _, P, C, t), move(_, P, C, t).                                        (9b)
```

encoding the domain axioms in Statements 4(c) and 2(d), which were (so far) unknown to the robot.

We also conducted these learning experiments starting with the same prior knowledge (Σ, statics, initial conditions, and non-target axioms), but using only examples of automated robot execution; the set of axioms were still learned successfully. In fact, even if the robot execution examples do not include all the rings and only represent incomplete examples, they provide sufficient information for the learn many of the axioms included in the original ASP representation of the domain. In addition, we tested the learned encoding (of axioms) in simulation in a scenario that mimics Figure 4. This scenario was chosen because of its complexity: all colored pegs are occupied, all four rings are in the scene, and transfer between the arms is only needed for some rings. The learned task specifications are sufficient to compute the correct plan for this scenario, establishing that the learned knowledge is accurate. Table 1 shows the speed of the learning process for each action specification, as well as the length of the learned axioms with respect to

Table 1: Quantitative results of the ILASP task.

Actions	Original axiom length	Learned axiom length	Learning time [s]
`move(Arm,ring,Color,t)`	4	4	0.487
`move(Arm,peg,Color,t)`	11	10	49.17
`move(Arm,center,t)`	5	5	0.093
`extract(Arm,ring,Color,t)`	2	2	0.058
`grasp(Arm,ring,Color,t)`	2	2	0.09
`release(Arm,ring,Color,t)`	2	2	0.786
total	**26**	**25**	**50.684**

the original ASP encoding of the domain. Results about the learning time refer to the learning process without any prior knowledge of the target axioms, which takes more time than learning with partial knowledge. We noticed that the action `move(Arm, peg, Color, t)` has the most influence on the results. In fact, learning identified a shorter representation for the original executability condition in Statement 4(a) by finding connections between related atoms. Moreover, `move(Arm, peg, Color, t)` has the highest number of related conditions, and more time is needed to find the correct set of hypotheses for it.

5. Conclusion

Autonomy in robots used for surgical tasks can provide substantial benefits, but it also requires the robot to reason reliably with incomplete domain knowledge and adapt to variations based on a limited number of training examples. Towards achieving this objective, the framework described in this paper combines the non-monotonic logical reasoning capability of Answer Set Programming (ASP) with the incremental learning capability of ILASP, a tool based on Inductive Logic Programming to learn knowledge encoded as ASP programs. As an illustrative example, we focused on peg transfer, an established and challenging surgical training task. Unlike the statistical learning algorithms typically used in similar scenarios surgical robotics, experiments indicate the ability of our framework to learn axioms encoding action preconditions and executability conditions based on only four human demonstrations and/or automated robot executions of the task. In addition, the ability to find semantic relations in contextual examples even helps reduce the complexity of the learned or manually encoded axioms.

Our framework represents a novel application of the principles of commonsense reasoning and inductive learning to surgical robotics; it opens up multiple directions for further research. In the future, we will evaluate our framework in more complex surgical scenarios, and enhance autonomy by exploring the learning of axioms encoding other kinds of knowledge [38]. Furthermore, we will build on the relational descriptions underlying the logic-based formalisms of our framework to make the robot's decisions more transparent, promoting reliability and trust in the robot's operation.

References

[1] Balduccini, M., 2007. Learning Action Descriptions with A-Prolog: Action Language C, in: AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning.

[2] Berthet-Rayne, P., Power, M., King, H., Yang, G.Z., 2016. Hubot: A three state human-robot collaborative framework for bimanual surgical tasks based on learned models, in: 2016 IEEE International Conference on Robotics and Automation (ICRA), IEEE. pp. 715–722.

[3] Blum, T., Padoy, N., Feußner, H., Navab, N., 2008. Modeling and online recognition of surgical phases using hidden markov models, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer. pp. 627–635.

[4] Camarillo, D.B., Krummel, T.M., Salisbury Jr, J.K., 2004. Robotic technology in surgery: past, present, and future. The American Journal of Surgery 188, 2–15.

[5] Charrière, K., Quellec, G., Lamard, M., Martiano, D., Cazuguel, G., Coatrieux, G., Cochener, B., 2017. Real-time analysis of cataract surgery videos using statistical models. Multimedia Tools and Applications 76, 22473–22491.

[6] Chen, J., Lau, H.Y., Xu, W., Ren, H., 2016. Towards transferring skills to flexible surgical robots with programming by demonstration and reinforcement learning, in: 2016 Eighth International Conference on Advanced Computational Intelligence (ICACI), IEEE. pp. 378–384.

[7] Corcione, F., Esposito, C., Cuccurullo, D., Settembre, A., Miranda, N., Amato, F., Pirozzi, F., Caiazzo, P., 2005. Advantages and limits of robot-assisted laparoscopic surgery: preliminary experience. Surgical Endoscopy and Other Interventional Techniques 19, 117–119.

[8] Cropper, A., Muggleton, S.H., 2019. Learning efficient logic programs. Machine Learning 108, 1063–1083.
[9] De Raedt, L., Kersting, K., 2008. Probabilistic inductive logic programming, in: Probabilistic Inductive Logic Programming. Springer, pp. 1–27.
[10] Dergachyova, O., Morandi, X., Jannin, P., 2018. Knowledge transfer for surgical activity prediction. International journal of computer assisted radiology and surgery 13, 1409–1417.
[11] Derossis, A.M., Fried, G.M., Sigman, H.H., Barkun, J.S., Meakins, J.L., 1998. Development of a model for training and evaluation of laparoscopic skills. The American journal of surgery 175, 482–487.
[12] DiPietro, R., Lea, C., Malpani, A., Ahmidi, N., Vedula, S.S., Lee, G.I., Lee, M.R., Hager, G.D., 2016. Recognizing surgical activities with recurrent neural networks, in: International conference on medical image computing and computer-assisted intervention, Springer. pp. 551–558.
[13] Erdem, E., Patoglu, V., 2018. Applications of ASP in Robotics. Kunstliche Intelligenz 32, 143–149.
[14] Estebanez, B., del Saz-Orozco, P., Rivas, I., Bauzano, E., Muñoz, V., Garcia-Morales, I., 2012. Maneuvers recognition in laparoscopic surgery: Artificial neural network and hidden markov model approaches, in: 2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob), IEEE. pp. 1164–1169.
[15] Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Thiele, S., 2008. A user's guide to gringo, clasp, clingo, and iclingo .
[16] Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T., 2012. Answer Set Solving in Practice, Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan Claypool Publishers.
[17] Gil, Y., 1994. Learning by Experimentation: Incremental Refinement of Incomplete Planning Domains, in: International Conference on Machine Learning, New Brunswick, USA. pp. 87–95.
[18] Ginesi, M., Meli, D., Roberti, A., Sansonetto, N., Fiorini, P., 2020. Autonomous task planning and situation awareness in robotic surgery. arXiv:2004.08911.
[19] Hong, M., Rozenblit, J.W., 2016. Modeling of a transfer task in computer assisted surgical training, in: Proceedings of the Modeling and Simulation in Medicine Symposium, pp. 1–6.
[20] Katzouris, N., Artikis, A., Paliouras, G., 2015. Incremental learning of event definitions with inductive logic programming. Machine Learning 100, 555–585.
[21] Laird, J.E., Gluck, K., Anderson, J., Forbus, K.D., Jenkins, O.C., Lebiere, C., Salvucci, D., Scheutz, M., Thomaz, A., Trafton, G., Wray, R.E., Mohan, S., Kirk, J.R., 2017. Interactive Task Learning. IEEE Intelligent Systems 32, 6–21.
[22] Lalys, F., Jannin, P., 2014. Surgical process modelling: a review. International journal of computer assisted radiology and surgery 9, 495–511.
[23] Law, M., 2018. Inductive learning of answer set programs. Ph.D. thesis. University of London.
[24] Law, M., Russo, A., Broda, K., 2016. Iterative learning of answer set programs from context dependent examples. Theory and Practice of Logic Programming 16, 834–848.
[25] Law, M., Russo, A., Broda, K., 2018. The complexity and generality of learning answer set programs. Artificial Intelligence 259, 110–146.
[26] Loukas, C., Georgiou, E., 2013. Surgical workflow analysis with gaussian mixture multivariate autoregressive (gmmar) models: a simulation study. Computer Aided Surgery 18, 47–62.
[27] Mack, M.J., 2001. Minimally Invasive and Robotic Surgery. Journal of American Medical Association 285, 568–572.
[28] Mizoguchi, F., Ohwada, H., Nishiyama, H., Yoshizawa, A., Iwasaki, H., 2015. Identifying driver's cognitive distraction using inductive logic programming, in: Proceedings of the 25th International Conference on Inductive Logic Programming (ILP '15).
[29] Mota, T., Sridharan, M., 2019. Commonsense Reasoning and Knowledge Acquisition to Guide Deep Learning on Robots, in: Robotics Science and Systems, Freiburg, Germany.
[30] Moustris, G.P., Hiridis, S.C., Deliparaschos, K., Konstantinidis, K., 2011. Evolution of autonomous and semi-autonomous robotic surgical systems: a review of the literature. The international journal of medical robotics and computer assisted surgery 7, 375–392.
[31] Muggleton, S., 1991. Inductive logic programming. New generation computing 8, 295–318.
[32] Neumuth, T., Strauß, G., Meixensberger, J., Lemke, H.U., Burgert, O., 2006. Acquisition of process descriptions from surgical interventions, in: International Conference on Database and Expert Systems Applications, Springer. pp. 602–611.
[33] Ng, R., Subrahmanian, V.S., 1992. Probabilistic logic programming. Information and computation 101, 150–201.
[34] Passerini, A., Frasconi, P., Raedt, L.D., 2006. Kernels on prolog proof trees: Statistical learning in the ilp setting. Journal of Machine Learning Research 7, 307–342.
[35] Reiley, C.E., Plaku, E., Hager, G.D., 2010. Motion generation of robotic surgical tasks: Learning from expert demonstrations, in: 2010 Annual international conference of the IEEE engineering in medicine and biology, IEEE. pp. 967–970.
[36] Sakama, C., Inoue, K., 2009. Brave induction: a logical framework for learning from incomplete information. Machine Learning 76, 3–35.
[37] Schüller, P., Benz, M., 2018. Best-effort inductive logic programming via fine-grained cost-based hypothesis generation. Machine Learning 107, 1141–1169.
[38] Sridharan, M., Meadows, B., 2018. Knowledge Representation and Interactive Learning of Domain Knowledge for Human-Robot Collaboration. Advances in Cognitive Systems 7, 77–96.
[39] Tao, L., Elhamifar, E., Khudanpur, S., Hager, G.D., Vidal, R., 2012. Sparse hidden markov models for surgical gesture classification and skill evaluation, in: International conference on information processing in computer-assisted interventions, Springer. pp. 167–177.
[40] Vidovszky, T.J., Smith, W., Ghosh, J., Ali, M.R., 2006. Robotic cholecystectomy: learning curve, advantages, and limitations. Journal of Surgical Research 136, 172–178.
[41] Yang, G.Z., Cambias, J., Cleary, K., Daimler, E., Drake, J., Dupont, P.E., Hata, N., Kazanzides, P., Martel, S., Patel, R.V., et al., 2017. Medical robotics—regulatory, ethical, and legal considerations for increasing levels of autonomy. Science Robotics 2, 8638.
[42] Zhang, Q., Li, B., 2011. Video-based motion expertise analysis in simulation-based surgical training using hierarchical dirichlet process hidden markov model, in: Proceedings of the 2011 international ACM workshop on Medical multimedia analysis and retrieval, pp. 19–24.