A Novel Parsimonious Cause-Effect Reasoning Algorithm for Robot Imitation and Plan Recognition

Garrett Katz, Di-Wei Huang, Theresa Hauge, Rodolphe Gentili, James Reggia

***Abstract*—Manually programming robots is difficult, impeding more widespread use of robotic systems. In response, efforts are being made to develop robots that use imitation learning. With such systems a robot learns by watching humans perform tasks. However, most imitation learning systems replicate a demonstrator's actions rather than obtaining a deeper understanding of why those actions occurred. Here we introduce an imitation learning framework based on causal reasoning that infers a demonstrator's intentions. As with imitation learning in people, our approach constructs an explanation for a demonstrator's actions, and generates a plan based on this explanation to carry out the same goals rather than trying to faithfully reproduce the demonstrator's precise motor actions. This enables generalization to new situations. We present novel causal inference algorithms for imitation learning and establish their soundness, completeness and complexity characteristics. Our approach is validated using a physical robot, which successfully learns and generalizes skills involving bimanual manipulation. Human performance on similar skills is reported. Computer experiments using the Monroe Plan Corpus further validate our approach. These results suggest that causal reasoning is an effective unifying principle for imitation learning. Our system provides a platform for exploring neural implementations of this principle in future work.**

***Index Terms*—artificial intelligence, imitation learning, cause-effect reasoning, abduction, parsimonious covering theory, cognitive robotics**

I. Introduction

MANUALLY programming contemporary robotic systems is time consuming, difficult, and expensive, requiring a trained roboticist to make changes for even slightly altered tasks. A potential alternative is to replace manual programming with *imitation learning*, in which a robotic system learns a task by watching a human perform the task, and then attempts to imitate what was observed. However, current methods for imitation learning are quite limited: The autonomous systems they produce typically focus on faithfully replicating the demonstrator's actions (e.g., arm trajectories) rather than "understanding" why those actions are occurring, and often do not generalize well even to mildly novel situations. We refer to faithful replication of arm trajectories as "low-level" imitation learning.

Empirical work from cognitive science and neuroscience conducted on humans and monkeys reveals that observation and imitation learning is an important component of our motor repertoire. This is in agreement with the idea that throughout the lifespan and in particular during early childhood development, humans and other primates gain procedural knowledge to a great extent through observation and imitation learning [29]. In particular, humans have neural mechanisms (the mirror neuron system) that likely facilitate decoding of intentions and understanding of actions previously observed in order to infer a forthcoming new goal [24]. Human imitation learning involves an understanding of the intentions of a demonstrator, in addition to their observed actions [3], [16], [20]. We refer to the understanding of a demonstrator's intentions as "high-level" or "cognitive-level" imitation learning.

Inferring a demonstrators's intent can be viewed as a form of cause-effect reasoning: How do hidden intentions cause the actions that are observed? Artificial intelligence (AI) researchers have extensively studied interpretive/explanatory *cause-effect reasoning*, also known as abductive inference, including its utility for inferring an agent's intentions [8], [28], [38]. This work has been applied to problems such as story understanding, human-computer discourse, and UNIX help systems, but the connection to robotic imitation learning is largely unexplored. Most past imitation learning research has focused on imitating sensorimotor control, with minimal examination of cognitive processing (e.g., [1], [2], [4], [15], [36], [43], [46], [50]). While certain cognitive abilities have been modeled in the context of imitation learning (e.g., [10], [13], [17], [25], [47]), to our knowledge the utility of causal reasoning in particular has not been studied in depth.

Based on such considerations, we hypothesize that causal reasoning is central to cognitive-level, general-purpose imitation learning. To test this hypothesis, we recently proposed an imitation learning framework in which causal reasoning plays a central role, shown in Fig. 1 [27]. In this current paper, we develop new algorithms to support this framework, establish formal guarantees of their correctness and efficiency, and demonstrate substantial improvements in performance using a broader array of experiments. Our approach constructs a *parsimonious explanation* for an observed demonstration, in which hypothesized intentions explain observed actions. Our algorithms incorporate ordering constraints and causal chaining: a high-level intention causes a sequence of sub-intentions, each of which cause their own sequence of sub-intentions, and so on, terminating with the sequence of observed actions. The high-level intentions can then be carried out in new situations that require significantly different low-level actions and motor plans (for example, if the robot has four arms instead of two). In other words, our goal is to create a system that

Garrett Katz is with the Comp. Sci. Dept. at Univ. of Maryland, College Park, 20742, USA, email: gkatz@cs.umd.edu
Di-Wei Huang is with the Comp. Sci. Dept. at Univ. of MD, College Park.
Theresa Hauge is with the Kinesiology Dept. at Univ. of MD, College Park.
Rodolphe Gentili is with the Kinesiology Dept., Neurosci. and Cog. Sci. Program, and MD Robotics Center at Univ. of MD, College Park.
James Reggia is with the Comp. Sci. Dept., Neurosci. and Cog. Sci. Program, MD Robotics Center, and Inst. for Advanced Comp. Studies at Univ. of MD, College Park.

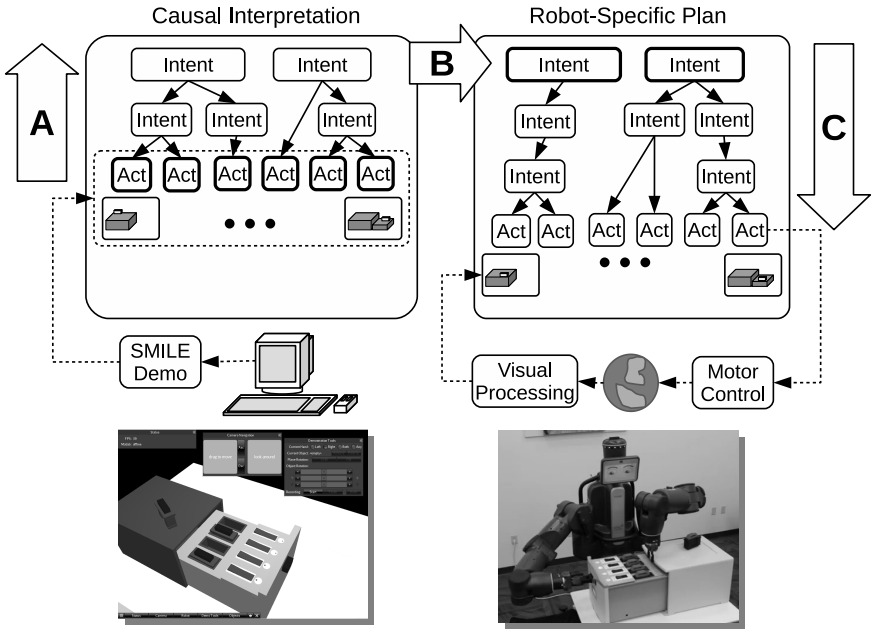

Fig. 1: An overview of our cause-effect imitation learning framework. Demonstrations are recorded in SMILE, a virtual environment (bottom left). SMILE output (dotted rectangle, top left) includes demonstrated actions (bold squares, top left) and resulting states of the environment (depicted as small 3D scenes, as shown in the dotted rectangle). Higher-level intentions that explain the demonstrated actions are then inferred by our system through causal reasoning (block arrow A). Solid line arrows indicate causal relationships between intentions and sub-intentions, forming a causal hierarchy, as pictured. When given a specific similar situation in the physical world, the inferred top-level intentions (bold rectangles, top right) represent an abstraction of the learned skill that can be reused for cognitive-level imitation by a robot in the new situation, by matching objects from the demonstration with objects in the new situation (block arrow B). Automated planning (block arrow C) is used to find new sub-intentions, actions, and ultimately motor plans (top right) that reflect the embodiment of the robot and carry out the same top-level intentions in a way more appropriate for the new situation. The planned actions are integrated with the robot's motor control and the new situation to drive execution of the skill in the physical world (bottom right), which may involve robot actions that differ from those of the human demonstrator, and which may include robot actions at a lower level of abstraction than those in the original demonstration. Visual processing is used to match objects in the new situation with corresponding objects from the demonstration and update the robot's internal model of the environment during execution.

can generalize based on a *single* demonstration. Since we are interested in the cognitive aspects of imitation learning, we take low-level sensorimotor processing as a given.[1] Our research group has conducted a complementary and parallel effort to develop a neuro-inspired sensorimotor model, that will be integrated into this framework in future work [18].

Our framework is validated empirically in a real-world application scenario, where a physical robot (Baxter, Rethink Robotics™) learns maintenance skills on a hard drive docking station. Demonstrations are recorded in a virtual environment called SMILE,[2] developed previously by our research group [22], [23]. Additional robotic experiments using toy blocks further validate the approach, and experiments with human participants inform the robotic framework. Our algorithms are also validated through an extensive battery of computer experiments using the well-known Monroe Plan Corpus testing set [5], a large dataset of 5000 automatically generated goals and associated plans, in a simulated emergency response domain. These experiments confirm the correctness and efficiency of our algorithms. The code for all causal reasoning algorithms and experiments is open source and freely available,[3] along with videos of demonstrations and subsequent robot imitation, details of the causal relations used in the evaluations of our approach, and details of the human experiments, in the Supplemental Online Information for this paper [42].

II. A Running Example: Learning Hard Drive Maintenance Tasks

To illustrate how our causal reasoning method works in a concrete fashion, we begin by considering a learning scenario that we call the "hard drive docking station." A robot must learn procedures for maintaining a docking station for several hard drives that are subject to hardware faults (Fig. 1, bottom left and right). Each drive slot is linked to an LED fault indicator and a switch that must be toggled before inserting or

[1]We use simple machine vision and motor control techniques that are less sophisticated than the state of the art but proved sufficient for our purposes.

[2]SMILE is available for download at https://github.com/dwhuang/SMILE.

[3]Code available at https://github.com/garrettkatz/copct.

removing drives. The goal is to replicate a teacher's *intentions*, based on just one demonstration, in new situations that require different motor plans. For example, if the teacher discards a faulty drive and replaces it with a spare, so must the robot, even when a different slot is faulty and the spare is somewhere else. Due to the robot's physical constraints, it may need to use different motor actions than those used by the demonstrator, such as using a different arm for certain steps, or handing off objects between grippers. For the purposes of experimentation, we used a mock-up docking station that we constructed (Fig. 1, lower right). The dock has faux 3D-printed "hard drives" and an Arduino controller for the LEDs and toggle switches.

To capture human demonstrations, we use SMILE, a virtual environment shown in Fig. 1 (bottom left). SMILE is a graphical computer program with intuitive GUI controls in which users can manipulate 3D objects located on a tabletop and record their actions. The recording is output in both video format and a machine-readable event transcript, describing which objects were grasped, with which hands, and real-time changes in object positions and orientations. Aside from the existence of a left and right hand, it contains no model of the user's embodiment (e.g., kinematic chains or joint angles), and no indication of the user's overarching intentions that lead them to grasp, move, and release various objects in various positions. Instead, objects are manipulated through mouse clicks, drag-and-drops, and keystrokes. As a result, SMILE bypasses the substantial image processing challenges of human motion capture, as well as the challenges of viewpoint transformation (including orientation, distance, and anthropometry) [34], [35].

III. Imitation Learning with Causal Inference

A. Learning Skills by Explaining Demonstrations

The basic intuition behind our approach is that a demonstrator mentally uses internal intentions at multiple levels of abstraction when demonstrating a new skill. The intentions at less abstract levels are more specific to the current situation in which the demonstration occurs, whereas the intentions at more abstract levels are more general to the skill being taught, and apply in multiple situations. In some sense, having an abstract intention *causes* an agent to have more concrete intentions relevant to the current situation. Therefore, causal reasoning can be used to explain a demonstrator's actions in terms of their abstract, general intentions.

To realize this intuition in practice, we rely on substantial background knowledge that must be compiled in a knowledge base by a human *domain author*. The three main types of information in the knowledge base are:

- **Objects:** The geometries, colors, part-whole relationships, and valid grasp postures for every type of object that might be present (e.g., hard drive, toggle switch, etc.).
- **Actions:** Action models that capture how the current state of the environment changes when the action is performed, reminiscent of STRIPS-style operators [14].
- **Causal Intention Relations:** Causal relationships between parent intentions and the sequences of child intentions they cause depending on the state of the environment, reminiscent of SHOP-style methods [31].

Given a demonstration event transcript from SMILE, our system draws on its knowledge base to infer higher-level intentions that can explain what was observed. All intentions are *parameterized*: An intention *schema* such as "`grasp`⟨*object*, *gripper*⟩" can be grounded by, for example, binding *object* to the value "drive 1" and *gripper* to the value "left," signifying that drive 1 is grasped with the left gripper. This representation conforms with the standards of AI planning, and is compact, since it relieves the system from explicitly storing every possible grounding in memory (there may be combinatorially many possible groundings when there are many objects in the workspace). Low-level intention sequences such as "grasp drive 1 with left," "move left above slot 3," "lower left," "open left" can be caused by higher-level intentions such as "insert drive 1 in slot 3 with left gripper," which in turn may be caused by intentions such as "get drive 1 to slot 3" (with any gripper), and so on. In our real-world robotics domain, real-valued parameters (omitted from the examples here for simplicity) are also needed to represent things like relative object positioning - for example, the precise target orientation of the left gripper above a slot, or the precise distance the left gripper should be lowered. Each individual intention, and the sub-intentions it can cause directly, must be pre-defined in the knowledge base provided by the domain author. These pre-defined units are the building blocks combined by our algorithms as they construct the complete explanations. The highest-level intentions pre-defined in our knowledge base for the dock maintenance scenario that we are using as an example here include dock manipulations such as "open dock" and "toggle switch," as well as a generic "get object 1 to object 2" intention, which may cause various sub-intentions such as temporarily emptying grippers and handing off objects between grippers, all depending on the intention parameters and the current state of the environment. Interpretation of demonstrations (Fig. 1, block arrow A) and planning for new situations (Fig. 1, block arrow C) both use logically equivalent forms of these causal relationships, but the former compute from effects to causes, while the latter compute from causes to effects.

Note that these high-level intentions are rather general, but not general enough that a single root intention can explain an entire demonstration. The typical demonstration can only be explained by a novel *sequence* of high-level intentions, which is not pre-defined in the knowledge base, and must be constructed through causal reasoning. One could use single pre-defined root intentions, but we elected not to in this work, in order to show that learning and generalization can be achieved by *constructing* new explanations through automated causal inference rather than *recognizing* existing explanations that were hand-coded in an exhaustive knowledge base.

The goal during learning is to infer the novel sequence of high-level intentions, in the correct order and with correct parameters, that explain the observed action sequence in a single demonstration. Our learning mechanism is a novel extension of *Parsimonious Covering Theory* (PCT), a formal computational model of cause-effect reasoning that has been applied in diverse fields such as medical diagnosis, circuit fault localization, natural language processing, and semantic web

technology [11], [21], [38], [49], but has not been applied to intention inference. PCT provides formal algorithms that create explanations (or "covers") for observed facts. Covers satisfying some notion of parsimony are considered most plausible. Various notions of parsimony have been formalized and evaluated in past work. The idea that parsimony is a unifying factor in explanation and inference is widely supported in philosophy and cognitive science (e.g., [9]).

In our context, a *cover* is a sequence of intentions that can explain all of the observed actions. The simplest covers containing the most general and abstract intentions are considered most *parsimonious*, and they are chosen as the robot's representation of a demonstrated procedural skill. This notion of parsimony is formalized in Sect. III-B.

In order to use PCT for this problem, we must model both causal chaining (i.e., intention A causes intention B, intention B causes intention C, etc.) and temporal ordering of effects (i.e., intention A causes an ordered sequence of sub-intentions: B1, then B2, then B3). While PCT has been extended separately for causal chaining and for temporal ordering [37], [48], it has not been extended to handle both simultaneously. Our main theoretical contribution is to extend PCT in this way. In addition, our method accommodates infinitely many causal associations, which is necessary for modeling intentions with real-valued parameters such as Cartesian positions, orientations, and relative spatial transformations describing objects in the current or target state of the environment. Thus our new causal reasoning algorithms constitute a rather general extension of PCT that is not limited to imitation learning but applies in any domain where causal chaining, temporal ordering, and spatial relationships are all important.

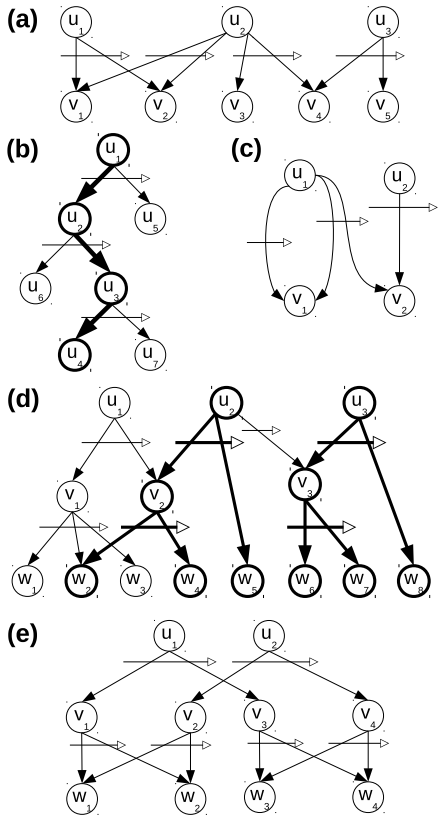

Fig. 2: Examples of causal relations. Nodes represent elements of V, vertical arrows represent causal relationships, and horizontal arrows represent ordering constraints. See text for further details.

B. Integrating Causal Chaining with Ordered Effects

We now present our formal model of causal chaining with ordered effects, which forms the core of our imitation learning method. To our knowledge, this is the first provably correct and efficient extension of PCT to handle both causal chaining and ordering constraints. This extension of PCT is used in our imitation learning framework to infer the intentions of a demonstrator based on their observed actions.

1) Formalizing Causal Knowledge: Let V be any (potentially infinite) set. Elements of V represent anything that can be a cause or effect: in our context, elements of V correspond to intentions, as pictured in the top half of Fig. 1. An ordered sequence of N elements from V is denoted $\langle v_i \rangle_{i=1}^{N} = \langle v_1, v_2, ..., v_N \rangle$. We denote the set of all such sequences using the Kleene star: V^*. For brevity, arbitrary sequences may be written with subscripts and superscripts omitted, as in $\langle v \rangle$, as long as we have not already used v in the current context to refer to an individual element of V.[4] When subscripts and superscripts are omitted, the length of an arbitrary sequence $\langle v \rangle$ is denoted $|\langle v \rangle|$. To denote a *sequence of sequences*, each member sequence is written with a parenthesized superscript: e.g., $\langle v \rangle^{(1)}, \langle v \rangle^{(2)}, ...$

[4]E.g., if we write "Given $v \in V$, consider $\langle v \rangle$ ...", we are referring to a length 1 sequence consisting of v, not an arbitrary sequence.

A *causal relation* over V is a set $C \subseteq V \times V^*$. An element $(u, \langle v \rangle) \in C$ signifies that u can *cause* the sequence $\langle v \rangle$. This means that u *may* cause $\langle v \rangle$, not that it *must*. However when u actually does cause $\langle v \rangle$ it must cause the full sequence in order. In our context, elements of C represent a parent intention causing a sequence of sub-intentions (downward arrows in Fig. 1). C can be depicted graphically by drawing elements of V as circles, with vertical arcs connecting causes with effects, and horizontal arrows through vertical arcs to indicate ordering constraints. Several examples shown in Fig. 2 are referenced throughout the following. Note that C may be many-to-many: the same u might cause any of several different $\langle v \rangle$'s and vice-versa. For example, in Fig. 2(a), $\langle v_1, v_2 \rangle$ can be caused by either u_1 or u_2. And u_2 can cause either $\langle v_1, v_2 \rangle$ or $\langle v_3, v_4 \rangle$.

Sequence membership is written with $\in$, as in $v_2 \in \langle v_1, v_2, v_3 \rangle$. Given any sequence

$$\langle (u_1, \langle v \rangle^{(1)}), (u_2, \langle v \rangle^{(2)}), ..., (u_\ell, \langle v \rangle^{(\ell)}) \rangle \in C^*,$$

if $u_{l+1} \in \langle v \rangle^{(l)}$ for all l from 1 to $\ell - 1$, then we refer to the sequence as a *causal chain* with depth ℓ. We restrict our attention to causal relations with no "loops": for any causal chain with depth ℓ, we assume $u_1 \notin \langle v \rangle^{(\ell)}$. An example of a causal chain is shown in Fig. 2(b) with the chain in bold: $\langle (u_1, \langle u_2, u_5 \rangle), (u_2, \langle u_6, u_3 \rangle), (u_3, \langle u_4, u_7 \rangle) \rangle$.

A *covering tree* is an ordered tree in which every ordered parent-child relationship, $(u, \langle v \rangle)$, is a member of C. In Fig. 1, each top-level intention is the root of the covering tree below it. The root of any covering tree is called a *singleton cover* of the ordered leaves. In Fig. 2(d), u_2 is a singleton cover of $\langle w_2, w_4, w_5 \rangle$, and u_3 is a singleton cover of $\langle w_6, w_7, w_8 \rangle$. The respective covering trees are shown in bold-face.

Consider I covering trees: u_1 is a singleton cover of $\langle w \rangle^{(1)}$, ... u_i is a singleton cover of $\langle w \rangle^{(i)}$, ... u_I is a singleton cover of $\langle w \rangle^{(I)}$. The sequence $\langle u_i \rangle_{i=1}^{I}$ is called a *cover* of $\langle w \rangle = \langle w \rangle^{(1)} \oplus ... \oplus \langle w \rangle^{(I)}$, where $\oplus$ denotes sequence concatenation. In other words, a cover is formed by (the roots of) an ordered forest of cover trees. Each u_i is referred to as a *singleton sub-cover* of $\langle w \rangle^{(i)}$. In Fig. 2(d), $\langle u_2, u_3 \rangle$ is a cover of $\langle w_2, w_4, w_5, w_6, w_7, w_8 \rangle$. Note that a singleton cover is simply a cover with a single element.

Let $\langle u \rangle$ be a cover of $\langle w \rangle$. If an associated covering forest has depth at most ℓ, then $\langle u \rangle$ is also called an *ℓ-cover* of $\langle w \rangle$. Any $\langle w \rangle$ is considered a 0-cover, or *trivial cover*, of itself.

The contiguous subsequence relation is written $\sqsubseteq$, as in $\langle v_2, v_3 \rangle \sqsubseteq \langle v_1, v_2, v_3, v_4 \rangle$. Given any $\langle w \rangle \in V^*$, suppose there is a non-empty subsequence $\langle v \rangle \sqsubseteq \langle w \rangle$ and a $u \in V$ such that $(u, \langle v \rangle) \in C$. In other words, suppose part of $\langle w \rangle$ can be caused by some u. Then $\langle w \rangle$ is referred to as *mid-level*. Otherwise, $\langle w \rangle$ is referred to as *top-level*. In Fig. 2(e), $\langle v_1, v_3 \rangle$ is a mid-level cover for $\langle w_1, w_2, w_3, w_4 \rangle$ because the (full) subsequence $\langle v_1, v_3 \rangle$ can be caused by u_1. In contrast, there are four distinct top-level covers of $\langle w_1, w_2, w_3, w_4 \rangle$: $\langle u_1 \rangle, \langle u_2 \rangle, \langle v_1, v_4 \rangle$, and $\langle v_2, v_3 \rangle$. Fig. 2(e) also shows that the roots of a top-level cover are not necessarily top-level nodes: while v_1 is *part of* a sequence caused by u_1, and v_4 is *part of* a sequence caused by u_2, there is no node that can cause the singleton sequence $\langle v_1 \rangle$, the singleton sequence $\langle v_4 \rangle$, or the sequence $\langle v_1, v_4 \rangle$. So $\langle v_1, v_4 \rangle$ is a top-level cover.

2) Formalizing Parsimonious Explanation: Two formal parsimony criteria used in PCT are *minimum cardinality* and *irredundancy*, although these do not involve temporal ordering in the original formulation [38]. In our context, $\langle u \rangle$ is a *minimum cardinality* cover of $\langle w \rangle$ if there is no other $\langle v \rangle$ with fewer elements that also covers $\langle w \rangle$. $\langle u \rangle$ is an *irredundant* cover of $\langle w \rangle$ if there is no proper subsequence $\langle v \rangle \sqsubseteq \langle u \rangle$ that also covers $\langle w \rangle$. Either criteria can be imposed depending on the problem domain, and only parsimonious covers are considered to be valid explanations. However, if minimum cardinality is imposed, irredundancy is automatically imposed as well, since a redundant cover has higher cardinality than the same cover with the redundant top-level node removed.

From the standpoint of imitation learning, our chief concern is identifying intentions that are as general as possible, so we propose an additional parsimony criterion: $\langle u \rangle$ is considered a *parsimonious* cover of $\langle w \rangle$ only if it is a *top-level* cover of $\langle w \rangle$. We impose minimum cardinality (and hence also irredundancy) in addition to top-level-ness in order to further reduce the set of valid explanations as necessary. Even if minimum cardinality were not imposed, top-level covers will generally satisfy the irredundancy criterion, since removing some roots from a covering forest will result in uncovered leaves.[5] Top-level-ness and irredundancy are also similar in spirit, since both emphasize covers that have been maximally simplified by local modifications: redundant covers are made irredundant by removing a subset; mid-level covers are made top-level by replacing a subsequence with its cause.

A *causal problem domain* (or simply "domain") is a pair $D = (V, C)$ where C is a causal relation over the set V. A *causal inference problem* is a pair $(D, \langle v \rangle)$, where D is a domain and $\langle v \rangle \in V^*$ is an observed sequence to be explained ($\langle v \rangle$ corresponds to an observed demonstration in imitation learning). The problem's *solution* is the set of all parsimonious (i.e., minimum-cardinality, top-level) covers of $\langle v \rangle$.

Several domain-specific constants are useful for quantifying the size of a domain:[6]

- M, the length of the longest sequence that can be caused by any $u \in V$ (i.e., $M = \sup_{(u,\langle v \rangle) \in C} |\langle v \rangle|$). $M = 3$ in Fig. 2(d), and $M = 2$ in Fig. 2(a),(b),(c), and (e). This refers to the length of an *ordered effect sequence* (i.e., "horizontally"), not a *causal chain* (i.e., "vertically").
- U, the largest possible number of distinct singleton covers of the same $\langle v \rangle$, taken over all $\langle v \rangle \in V^*$. That is,
$$U = \sup_{\langle v \rangle \in V^*} |\{u \in V \mid \langle u \rangle \text{ covers } \langle v \rangle\}|.$$
In Fig. 2(d), $\langle w_6, w_7 \rangle$ has two distinct singleton covers: u_2 and v_3. All other possible node sequences have 2, 1, or 0 distinct singleton covers, so $U = 2$.
- L, the depth of the deepest causal chain (i.e., $L = \sup_{\gamma \in \Gamma} |\gamma|$, where $\Gamma \subset C^*$ is the set of all causal chains). $L = 1$ in Fig. 2(a) and (c), $L = 2$ in Fig. 2(d) and (e), and $L = 3$ in Fig. 2(b).

Although we allow V (and hence C) to be infinite, we restrict our attention to domains where the constants M, U, and L are all finite.

3) Parsimonious Covering Algorithms: Our main technical contribution is a set of algorithms that solve the Parsimonious Covering problem as defined above. Specifically, given the background knowledge stored in C, and an observed sequence of effects (e.g., a plan of actions), the algorithms compute every parsimonious explanation of the observations (e.g., a sequence of hidden intentions). In Fig. 1, these algorithms correspond to block arrow A, in which the top-level intentions are inferred based on actions recorded in SMILE. The algorithms are provably sound: any $\langle u \rangle$ computed by the algorithms is guaranteed to be a true top-level cover of $\langle w \rangle$. They are also provably complete: any true top-level cover $\langle u \rangle$ of $\langle w \rangle$ is guaranteed to be found by the algorithms. The algorithm outputs can be filtered for minimum cardinality as a post-processing step. The algorithms can also be shown to be fixed-parameter tractable in M, the bound defined above, and hence they can be run in a reasonable time frame on the vast majority of problem instances, as borne out by experiment in Sect. IV.

[5]There are contrived exceptions: In Fig. 2(c), $\langle u_1, u_2 \rangle$ covers $\langle v_1, v_2 \rangle$, but is redundant, since $\langle u_1 \rangle$ also covers $\langle v_1, v_2 \rangle$. Causal relations like this do not occur in our imitation learning domain and are rare in the Monroe Plan Corpus, so we consider them pathological in practice.

[6]In the following, sup refers to the mathematical supremum; i.e., the least upper bound of some quantity over a given set. If the sets were finite, the supremum would be equivalent to the maximum.

Full pseudo-code, algorithmic descriptions, and proofs are given in the appendices. Here we summarize the key ideas. Given a causal inference problem, the solution (i.e., the set of all top-level covers) is computed by an algorithm we call EXPLAIN (see Appendix A, Fig. 5). The main input to EXPLAIN is the background knowledge contained in C, and the sequence of observations $\langle w \rangle$. The background knowledge is represented in EXPLAIN as a function called CAUSES, which is defined as follows:

$$\text{CAUSES}(\langle v \rangle) \stackrel{\text{def}}{=} \{u \in V \mid (u, \langle v \rangle) \in C\}.$$

In other words, CAUSES returns all singleton 1-covers of its input. The domain author is responsible for correctly implementing CAUSES. It constitutes the interface between EXPLAIN and the causal knowledge base, and EXPLAIN treats CAUSES as a black box. Internally, EXPLAIN operates in two phases, which are described in detail in Appendix A. The first phase is a dynamic programming routine that decomposes the original problem into sub-problems and solves the sub-problems in a bottom-up fashion. The second phase reconstructs the solution to the original problem by recursively combining the solutions to the sub-problems found in the first phase.

In implementing CAUSES, the domain author only needs to enumerate the *direct* causal associations $(u, \langle v \rangle)$ that make up C. EXPLAIN automates the work of *composing* causal associations over multiple layers of causation and multiple sequences of effects. In other words, the domain author is only responsible for specifying the immediate parent-child relationships that are allowed, not for anticipating any of the multi-layer trees that might need to be constructed to form a covering forest. For example, in the upper left of Fig. 1, the domain author must specify that the first two actions on the left can be caused by the mid-level intention directly above them, and that the first two mid-level intentions on left can be caused by the intention above them, and so on, but need not specify that the full sequence of 6 low-level actions can be indirectly caused by the full sequence of the 2 top-level intentions. As such, EXPLAIN is a generic, domain-independent algorithm, that can find parsimonious, high-level covers for long, low-level sequences that need not be anticipated by the domain author or built in to the background knowledge.

C. Formalizing Intentions for Imitation Learning

While EXPLAIN is potentially useful in any domain involving causal chaining and ordering constraints, our chief focus in this work is intention inference and imitation learning. In this section we describe one way to encode the intention inference problem using the formal apparatus defined above.

Parameterized intentions can be formalized for a causal problem domain as follows. We let X be the set of all possible parameter values, including gripper identifiers, object identifiers, position vectors and orientation matrices, transformation matrices, joint speeds and tolerances, and so on. We let S be the set of all possible states of the robot's environment. Each state $s \in S$ includes current joint angles and gripper openings, a listing of symbolic relationships between object identifiers that hold in the current state such as "`is-on(block1, block2)`," and a mapping from each object identifier to geometric descriptors of the object's shape, position, and orientation in the current state. Lastly, we let $\mathcal{I}$ be the set of all intention *schemas*, which are intentions without parameters bound to any values (e.g., "`grasp`$\langle$*object*, *gripper*$\rangle$," where *object* and *gripper* are left unspecified).

Let $V = S \times \mathcal{I} \times X^*$. This is the V used in the causal problem domain. Each $v = (s, t, \langle x \rangle) \in V$ signifies an intention $t \in \mathcal{I}$ with parameter list $\langle x \rangle \in X^*$ in a state $s \in S$. Including states and parameters is important since the same intention may cause different sub-intentions depending on the input parameters and the current state.

A distinguished subset $A \subset V$ represents the *observable actions*. These are the concrete intentions such as grasp and release that are directly observable in a SMILE demonstration. The demonstration is formally a sequence $\langle a \rangle \in A$. This is the sequence to be explained and passed as input to EXPLAIN.

D. Encoding Causal Background Knowledge

Here we provide more detail on our implementation of CAUSES for the imitation learning domain, in order to clarify how much background knowledge must be built into the system, and to serve as a case study for one way a domain author might implement CAUSES. Both dock maintenance and block stacking below use the same causal relation (and the same CAUSES function). Some of the included intentions cannot be sensibly applied to blocks, such as `open-dock` and `toggle-switch`, but all remaining generic intentions such as `move-object` or `hand-off` can be applied to any objects including hard drives and blocks. The full set of causal relations are available as Supplemental Online Information [42]. Here we detail the most important examples.

At the lowest level of the causal hierarchy are sensorimotor primitives suitable for robot execution: opening or closing grippers, moving the actuators through a trajectory of joint angles, and detecting objects in raw visual data. The implementation of these capabilities is based on standard methods in robotics and beyond the scope of this paper. One level higher, joint trajectory actions can be caused by an intention to locate the end effector at a certain position and orientation. This causal relationship can be implemented by computing the robot's forward kinematics, given the last set of joint angles in the trajectory, to infer the intended placement of the end effector. In turn, the intention to locate an end-effector with a given placement can be caused by a slightly higher-level intention to locate a gripped object at a given placement. This causal relationship involves a coordinate transformation between the coordinate frame of a gripper itself and the coordinate frame of an object being gripped, which can differ by several centimeters in translation and substantial angular rotations depending on where the object was grasped. The actions observed in SMILE map directly onto grasp, release, and gripped object placement intentions, with minimal reformatting of the SMILE event transcript.[7]

[7]The very lowest primitives in the causal relation, namely joint angle trajectories and visual processing, are in fact at a lower level than what is observed in SMILE, and while they do not factor into intention inference, they come into play during robot planning and execution.

At a slightly higher level of abstraction, there are intentions to pick-up and put down objects with specific grippers. A pick-up intention will cause two of the sub-intentions described above: positioning a specified end-effector at a valid grasp affordance for the object, and then closing the gripper. The valid grasps for each type of object are included as built-in knowledge. The parameters for a put-down intention include the target position and orientation where the object should be put down. This also causes two sub-intentions: a gripped object placement, followed by a gripper opening. Other intentions that can cause end effector placements and gripper changes include hand-offs between grippers, pressing dock toggle switches, and opening/closing the dock.

At the next level up is an intention to move an object from "point A" to "point B," assuming that the object to be moved is unobstructed, and that the grippers needed are not already holding anything else. This intention may cause one of several combinations of pick-ups, inter-gripper hand-offs, and put-downs, depending on which grippers can reach the source location and which can reach the destination. One more level up, there is an intention that does not assume the grippers are already empty, but causes the necessary put-downs to empty the grippers before causing the move object sub-intention.

All of this knowledge must be encoded as relationships between various parent and child intentions, each of the form $(s, t, \langle x \rangle) \in S \times \mathcal{I} \times X^*$. The states and parameters of each parent are generally non-trivial functions of the states and parameters in the associated child sequence, which can involve geometric transformations and other continuous-valued mathematics. However, the parent and child *schemas* involved in each relationship are symbolic and generally conform to one of a relatively small set of *templates*. For example, the "move-object" parent schema can cause the child schema sequence ⟨"pickup," "hand-off," "putdown"⟩, where the mapping between child and parent states and parameters requires continuous computations, but the mapping between child and parent schema is discrete. More formally, the full set of causal knowledge conforms to a relatively small list of templates of the form $(\hat{t}, \langle \tilde{t} \rangle)$, where each template associates a parent schema $\hat{t}$ with one of the child schema sequences $\langle \tilde{t} \rangle$ that it can cause. When CAUSES is invoked on an arbitrary input sequence $\langle v_i \rangle_{i=1}^m$, where each v_i has the form $(s_i, t_i, \langle x \rangle^{(i)})$, the input schema sequence $\langle t_i \rangle_{i=1}^m$ is matched against each of the template effect sequences $\langle \tilde{t} \rangle$. Whenever there is a match, it means that the associated parent schema $\hat{t}$ could be a valid cause of the input schema $\langle t_i \rangle_{i=1}^m$, as long as the accompanying states and parameters are correctly related. So the corresponding states s_i and parameters $\langle x \rangle^{(i)}$ in the children are transformed as appropriate to populate the valid state $\hat{s}$ and parameters $\langle \hat{x} \rangle$ for the parent schema $\hat{t}$. These transformations are implemented by the domain author in the body of the CAUSES function. The resulting grounded parent intention, $(\hat{s}, \hat{t}, \langle \hat{x} \rangle)$, is included in the output of CAUSES.

In sum, a rather large set of background knowledge must be provided to the system up front, in order to handle the continuous-valued computation and complexity needed for real-world robotics. However, while domain authors must provide substantial details regarding the *direct* causal relationships, they need not anticipate all possible *indirect* causal chains that can occur over multiple links of causal chaining, or all possible action sequences that occur when several intentions are carried out in succession. For example, the longest effect sequence built into C comes from the "hand-off" intention, which can directly cause a sequence of 6 interleaved end-effector placements and gripper opening/closings. But the longest demonstration sequence we tested (see Sect. IV) was a block stacking task with 39 gripper operations, end effector placements, and gripped object placements. It would be uninteresting (and infeasible) for the domain author to anticipate every such sequence. Instead, EXPLAIN does the heavy lifting of inferring all parsimonious top-level intention sequences that can cover the full demonstration. Moreover, since CAUSES encapsulates the continous-valued mathematics, EXPLAIN does not need to explicitly reason about geometry or kinematics. As long as CAUSES correctly computes the parameters in its output, the correct values will be received by EXPLAIN every time it invokes CAUSES, and eventually propogated to the roots of the covering forests as they are constructed bottom-up.

E. Post-Learning Imitation and Generalization

Every time a new demonstration is interpreted, the inferred parsimonious covers are added to a growing library of learned skills, and the robot can be asked to imitate any of them at any time. So the system can be (re)programmed to perform different imitation tasks after others have already been learned, without losing any learned knowledge that has been acquired previously. Once the most parsimonious covers for an observed demonstration have been found and saved in this library (Fig. 1, upper left), the robot is ready to apply and generalize the learned skill to new situations (Fig. 1, upper right). These new situations are real, physical situations which need to be interpreted by the physical robot's visual system.

Consequently, when asked to imitate (i.e., after learning), the robot begins with visual processing to detect objects in the new situation, producing an internal representation $s \in S$ of the initial state. Since advanced image processing is not a focus of this work, our current implementation uses simple computer vision techniques. The objects found by visual processing must be matched with the corresponding objects in the original demonstration. For example, consider a dock maintenance scenario with three drives, two of which get manipulated by the demonstrator. When the robot sees three drives in a new situation, it does not know a priori which one should be treated as "drive 1" from the demonstration, which should be treated as "drive 2", and which is extraneous. We allow that different numbers of objects may be present, as long as each object manipulated in the demonstration has a counterpart in the new situation. For example, the robot will not be expected to imitate a demonstration of replacing a faulty drive with a spare drive in a situation where no spare drives are present.

The demonstration-object to real-world-object matching procedure we employed in this work is based on the premise that object colors, shapes, and part-whole relationships are more important than spatial positions (this could, of course,

be changed). This is because the robot should be able to generalize to new situations where the initial object positions are completely different than those observed in the demonstration. Part-whole relationships are predefined so that functionally related units are grouped. For example, in the dock maintenance scenario, each drive slot is grouped with its associated toggle and LED indicator as parts of a "dock module" object. The four dock modules are then grouped as parts of the dock drawer. The dock drawer and dock casing are then grouped as parts of the dock. The matching procedure is designed recursively in order to accommodate the trees of part-whole object relations.[8] Each level of recursion compares a demonstration sub-tree with a real-world sub-tree. At the deepest levels, the leaves are compared based on shape and color and a quantitative similarity measure is passed up the recursion. At the shallower levels, each child of the root of a demonstration sub-tree is recursively compared with each child of the root of a real-world sub-tree. The similarity measures recursively computed for each child pairing are used as weights for a weighted bipartite matching, which produces the optimal pairing of demonstration children with real-world children. The summed similarities across this optimal pairing are then passed up the recursion. By design, the results of this procedure are such that in the dock example they will match dock modules that are similar with respect to LED colors and slot occupancies, but potentially dissimilar with respect to their position on the dock drawer. For example, suppose that in the demonstration, slot $1^{(demo)}$ is occupied and LED $1^{(demo)}$ is red, and in the new scene, slots $2^{(new)}$ & $3^{(new)}$ are occupied but only LED $3^{(new)}$ is red. Slots and LEDs $1^{(demo)}$ & $3^{(new)}$ will be matched, rather than $1^{(demo)}$ & $2^{(new)}$, since the configuration of colors, shapes, and part-whole relationships is better preserved. The matching only compares the initial state in the demonstration with the initial state during imitation. Using inferred intentions while matching is a potential research direction, although similar problems involving plan reuse are known to be hard [33].

Once matching is complete, variable substitution is used to update parameters in the top-level intentions so that object identifiers from the demonstration are replaced with the matched identifiers in the real world. AI planning techniques are then used in a top-down manner to plan a sequence of low-level motor commands that carry out these learned intentions (see Fig. 1). We currently employ *Hierarchical Task Network* (HTN) planning, in which high-level tasks are decomposed into lower-level sub-tasks and ultimately executable actions [19]. It turns out that intentions in our causal hierarchy can be mapped directly onto the formal HTN notion of tasks. A corollary is that, if CAUSES formally inverts the HTN planning operators, then EXPLAIN formally inverts the HTN planning algorithm. To our knowledge, this is the first provably correct inversion of HTN planning, as established in Appendix B.

Like the causal relation, HTN operators can map the same parent intention onto several alternative child sequences. These represent alternate strategies for carrying out the parent intention, some of which may be more or less appropriate depending on the current state of the environment, and some of which may be quite different from what was actually done by the human demonstrator. The HTN planner can search each branch, simulating its effects on the environment, and avoid branches that fail. Consequently, the resulting actions planned for the new situation may differ significantly from the observed actions in the demonstration, as illustrated schematically in Fig. 1. For example, suppose the demonstrator swapped two hard drives by removing one with the left hand, the other with the right hand, and then inserting them back in opposite slots, but the slots are difficult for the robot to reach with its left hand due to physical constraints. When the HTN planner searches the branch of the high-level `move-object` task that uses the left arm, motion planning fails and the search must backtrack. An alternate search branch succeeds where the right gripper removes the first drive, hands it to the left which stages it on top of the dock, moves the second drive to the slot that initially contained the first drive, and finally moves the first drive to the slot that initially contained the second. This is an example of the bimanual coordination supported in our implementation.

The robot's capacity for generalization boils down to the synergistic combination of causal inference, object matching, and planning (arrows A, B, and C in Fig. 1). Object matching ensures that intentions are applied to the correct objects. Causal inference ensures that the most plausible intentions are applied, and planning ensures that they are applied in a way that respects the embodiment of the robot, rather than the human demonstration. Causal reasoning is a crucial component because it identifies intentions that are as high-level as possible given the pre-defined background knowledge. Higher-level intentions are better for generalization because the same parent intention can cause any of several alternative sub-intention branches, in a way sensitive to the current state of the environment. The results of just one branch are observed in the demonstration, but many other branches exist that are more appropriate for other situations. Inferring the demonstrator's intentions exposes these other branches, and the higher up the hierarchy, the more branches get exposed. So cause-effect inference of parsimonious covers is central to generalization after learning from a single demonstration. Moreover, the lowest-level HTN operators can invoke motion planning routines, which convert target gripper positions into joint angles that respect the physical constraints of the robot. As a result, the causal hierarchy can extend deeper than the actions recorded in SMILE, producing concrete motor plans suitable for physical robot execution (illustrated schematically in Fig. 1 by the deeper tree on the right-hand side).

IV. Experimental Results

We conducted two sets of experiments, one involving the full imitation learning and execution pipeline, and one focused on our causal inference algorithms in particular.

A. Imitation Learning Trials

With regards to robot imitation learning and execution as a whole, we were interested in two questions: How quickly can a robot interpret and imitate a demonstration? How often

[8]Part-whole trees are unrelated to the covering trees defined previously.

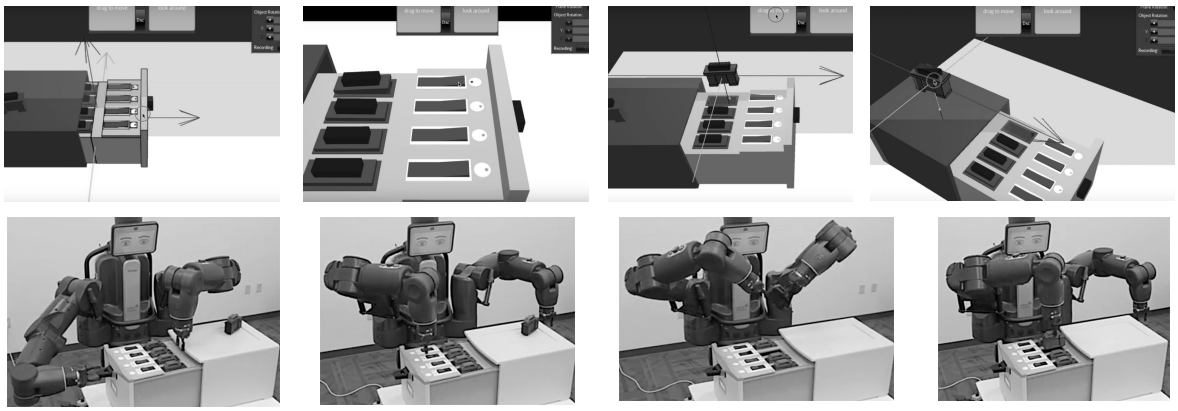

Fig. 3: Screenshots from a task demonstration in SMILE (top row) and the corresponding subsequent robotic imitation of the task (bottom row). Time flows from left to right. During imitation, a different LED is red corresponding to a different drive that should be replaced than the one observed in SMILE, and the robot correctly generalizes to this situation. Also, among other instances of bimanual coordination, the robot generalizes by performing a hand-off required by its embodiment that was not demonstrated in SMILE (bottom row, third panel from left). Full videos of the demonstration and robotic imitation are available as Supplemental Online Information [42].

and at what points in the process does it fail? To answer these questions, we ran a series of imitation learning trials using the dock maintenance scenario and a toy block stacking scenario. The tasks used in these trials are intentionally simple, intended solely to establish that EXPLAIN can work effectively in a real-world robotics application.

First, for the dock maintenance scenario, in each trial, the Baxter robot interpreted a demonstration of a maintenance skill, and then imitated that skill in a new situation, where the initial configuration of drives, slots, and LED indicators was different than what had been observed in the demonstration. Each trial was timed, and successes and failures were recorded.

More specifically, four different skills were taught to the robot: (1) discarding a red LED drive, (2) replacing (and discarding) a red LED drive with a spare on top of the dock, (3) replacing (and discarding) a red LED drive with a green one, and (4) swapping a red LED drive with a green one. Each skill was demonstrated twice in SMILE, using different initial states for the dock each time, to introduce more variety into our testing set. EXPLAIN was used to infer intentions in each demonstration (i.e., learning was always done with a single demonstration). Finally, the robot was asked to imitate each demonstration 4 times, again using different initial physical dock states each time. The result is 8 demonstrations total and 32 imitation trials total (8 demonstrations for learning $\times$ 4 new initial states per demonstration for imitation). In every demonstration and trial, the initial dock states were automatically and randomly generated, varying the number and position of spare drives, which slots were occupied, and which LEDs were red. Fig. 3 depicts a SMILE demonstration and the robot imitating it.

Second, in the toy block scenario, blocks were stacked to form letters of the English alphabet. Three demonstrations were recorded in SMILE: (1) Forming "IL," (2) forming "AI," and (3) forming "UM." Again, the robot was asked to imitate each demonstration 4 times, using different random initial block placements each time. The result is 12 additional trials. In both dock and block trials, every trial is independent from the others: The robot does not use other demonstrations or previous trials to improve its performance, it always imitates from scratch using a single demonstration.

To verify that these skills are reasonable targets to be learned from a single demonstration, we had a small sample of human participants ($n = 5$) perform similar tasks. We found that about 85% of the time they would correctly perform these tasks after observing only a single demonstration of the task. Further details of these human experiments can also be found in the Supplemental Online Information [42].

The first step of robotic imitation is interpretation of the demonstration through causal inference (arrow A in Fig. 1). This step is performed by EXPLAIN. Table I shows the results of running EXPLAIN on every demonstration. N is the size of the input, i.e., the number of discrete steps (grasps, releases, toggle switches) recorded in the SMILE event transcript. Runtime is the running time in seconds of EXPLAIN.[9] TL indicates the number of top-level covers found. MC indicates the number of minimal cardinality top-level covers found.

On every demonstration, EXPLAIN terminated in under 1 second (Table I), so time complexity was very reasonable in practice, at least for the simple tasks that we used here. This is especially acceptable given that intention inference is only required once per demonstration, and then immediately allows generalization to a variety of new situations. We also found

[9]Run times were measured on one 3.5GHz Intel Xeon CPU core. To account for noise from multi-threading with unrelated jobs on the workstation, the minimum run time from 10 repetitions is reported. Minimums are generally regarded as preferable to averages when empirically measuring run time, since they approximate the idealized run time when the operating system does not interleave any unrelated processes on the CPU. Run times for the same experiment on a laptop with a 2.4GHz Intel Core i7 CPU were about 1.5$\times$ as long.

TABLE I: EXPLAIN Performance

Demonstration	N	Runtime	TL	MC
Remove red drive (1)	7	0.008s	4	2
Remove red drive (2)	10	0.007s	4	2
Replace red with green (1)	15	0.024s	4	1
Replace red with green (2)	15	0.024s	4	1
Replace red with spare (1)	14	0.021s	8	2
Replace red with spare (2)	14	0.020s	4	1
Swap red with green (1)	16	0.017s	2	1
Swap red with green (2)	16	0.017s	2	1
Toy blocks (IL)	18	0.039s	64	1
Toy blocks (AI)	21	0.064s	128	1
Toy blocks (UM)	39	0.834s	8192	1

that while the number of top-level covers can become rather large, pruning by minimum cardinality was sufficient to nearly uniquely determine a suitable cover.[10]

After EXPLAIN processed each demonstration, an arbitrary minimum cardinality top-level cover was selected as the robot's representation of the learned skill. This cover was then used for imitation (i.e., object matching, HTN planning, and physical robot execution) in the new initial conditions for the current trial. Manual inspection confirmed that in all dock and block trials, the robot was generating a suitable, correct plan of low-level actions to execute. The plan was correct in that, barring sensorimotor errors such as failed grasps, the planned actions would accomplish the skill that had been demonstrated. Sample videos of a SMILE demonstration and the corresponding robotic execution of the learned skill can be found in our Supplemental Online Information [42].

Unfortunately, our physical robot failed mid-way through execution in 3 of the 32 dock maintenance trials ($\sim$9%) due to sensorimotor errors. In one failed trial, a drive was poorly aligned with a slot, and became stuck halfway down during insertion. In another, a drive was dropped during a hand-off, and in the third, a drive on top of the dock was knocked over during an arm motion. These issues are due to a combination of our simplistic sensorimotor processing routines and limited accuracy in Baxter's motor control as compared to more expensive robots. Nevertheless, the key result is that the cognitive learning process and subsequent imitation plans were correct in 100% of the trials. Since sensorimotor processing is not our primary focus here, we do not consider the execution fail rate of the physical robot to be a significant objection to the central goal of this work (although we are currently working to improve the sensorimotor processing).

In the block stacking trials, since each letter is made of three to eight blocks, these demonstrations tended to involve more steps than dock maintenance and took longer for EXPLAIN to process. The increased number of steps also results in combinatorially more top-level covers, as evident in Table I. Fortunately, minimum cardinality was an effective parsimony criterion for mitigating this effect. Also, although block stacking is conceptually simpler than dock maintenance, it poses a greater sensorimotor challenge, because the blocks are smaller than the drives and leave less room for error in visual processing and motor control. Stacking multiple blocks is also more difficult since small noise in the placement of a bottom block makes successful placement of the blocks above less likely. Lastly, the increased number of actions required to complete the task presents more opportunities for sensorimotor errors. We found that across all block stacking trials, 97% of the individual pick and place block movements were successful, 87% of the individual letters were successfully constructed, and 75% of the trials were successfully completed in full.

B. Monroe Plan Corpus Experiments

We ran a more extensive battery of experiments[11] on the much more complex Monroe Plan Corpus (MPC), a well-known benchmark from the field of plan recognition [5], to systematically assess the performance of EXPLAIN. We were interested in several questions: Are our theoretical results consistent with empirical evidence in a larger, more complex problem domain? What is the empirical average case complexity of EXPLAIN? Is EXPLAIN effective on problem domains we did not devise ourselves? These trials did not involve the use of a physical robot nor plan execution; they only measure the performance of EXPLAIN during intention learning.

The MPC is based on an HTN planning domain for emergency response in Monroe County, New York. Top-level goals, such as clearing a car wreck or repairing a power line, are accomplished through sequences of lower-level tasks such as navigating a snow plow or calling a power company. The corpus consists of 5000 planning examples, each of which is the HTN plan tree for a randomly chosen top-level goal in a randomly generated initial state. We refer to the top-level goal in each example as the "original" or "ground-truth" top-level goal. The low-level actions in each plan tree served as the input "demonstration" to EXPLAIN (the ground-truth top-level goal and intermediate tree nodes were withheld). The ground-truth top-level goal served as the target output that EXPLAIN was expected to compute.

In order to use EXPLAIN on this corpus, we needed to implement a CAUSES function. This implementation is nearly identical to the original HTN planning domain written by the creator of the corpus, with case-by-case logic for each planning operator, the only difference being that causes (i.e., parent tasks) are computed from their effects (i.e., child tasks) and not vice versa. The full set of causal relations are described in the Supplemental Online Information [42].

Another issue is that the MPC does not retain the initial or intermediate states present when the example was generated. Only the planned tasks and actions are included, without any state information. However, our formalization of intentions as described in Sect. III-C expects state information to be included, because some causes cannot be uniquely determined without it. The MPC is no exception. For example, there

[10]In the cases where there were two minimum cardinality covers, the difference was inconsequential: for example, a `discard-object` intention in one of them would be replaced by a `move-unobstructed-object-to-free-spot` intention, where both moved the same object, and the latter had the discard bin as the destination on which a free spot was found. This could be attributed to a design fluke in our definition of the robot's domain knowledge.

[11]The remaining experiments reported below were all run on a workstation with twelve 3.5GHz Intel Xeon CPU cores and 32GB of RAM, taking approximately three days in total.

is a task `(clean-up-hazard ?from ?to)` where variables `?from` and `?to` are locations specifying a road with hazardous conditions. This task may cause a single action `(!call fema)`. Since the parent parameters do not occur in the child, they cannot be inferred solely from the `!call` action. One possible work-around is to enumerate every possible `?from`, `?to` pair in the domain, and return every possible pairing in the output of CAUSES. This will lead to the correct explanation but also many other explanations, reducing the precision of EXPLAIN. This is compounded combinatorially when several such instances occur in sequence. In contrast, inspecting the state can reveal which road was hazardous, enabling CAUSES to only return the correct parent and no others. Fortunately, it is possible to partially reconstruct the states using an automated procedure which cross-references the planning operators in the domain definition [42]. We applied this procedure to every example in the corpus. Each partially reconstructed state was paired with its corresponding low-level action, according to the formalization described in Sect. III-C, before being passed as input to EXPLAIN.

We tested EXPLAIN on each and every example in the corpus as follows. We withheld the top-level goal (and plan tree), providing only the low-level state and action sequence as input. The output of EXPLAIN, $tlcovs$, is a set of top-level covers for the actions. To gauge correctness, we check that $tlcovs$ includes the ground-truth top-level goal. To gauge precision, we count how many other top-level covers are also included. We also record running time of EXPLAIN on each test example. If EXPLAIN was still running after 10 minutes, it was terminated early. This occurred in 165 of the 5000 examples (3.3%), which are excluded from the results reported below. Aside from this small portion of the dataset, run time was generally quite reasonable: On the 4835 examples allowed to run to completion, EXPLAIN required an average of 20 seconds. We found that $tlcovs$ included the original top-level goal in 4793 testing examples out of the 4835 that did not time out ($\sim$99.1% correct).[12]

On the other hand, $tlcovs$ was very imprecise, often containing more than 1000 possible explanations. The combinatorial explosion of top-level covers occurred because some MPC plan operators are designed such that many different parameterizations of a parent task can cause the same child task. When this occurs multiple times in a sequence, the number of valid covers at the next layer up grows exponentially. By pruning $tlcovs$ with the additional parsimony criterion of minimum cardinality, precision was substantially improved: in 3892 examples of the 4835 that ran to completion ($\sim$80%), there was only one minimum cardinality top-level cover (the ground-truth one); in 4351 examples ($\sim$90%), there were at most 12. This improvement is illustrated in Fig. 4.

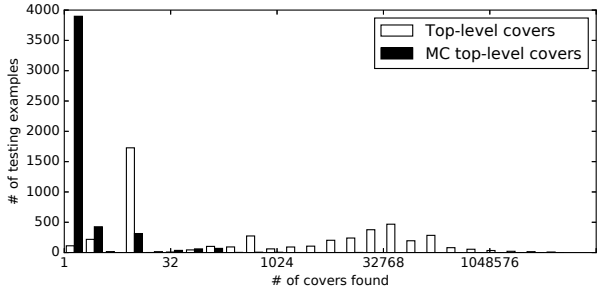

Fig. 4: Histogram comparing the precision of $tlcovs$ before and after pruning based on minimum cardinality.

Pruning $tlcovs$ by minimum cardinality did not come with any price to correctness either: this is because every example in the corpus is generated from a single top-level task. Since all ground-truth top-level covers are singletons, it is necessarily true that they will be included in the minimum cardinality subsets of $tlcovs$. By the same reasoning, all ground-truth top-level covers will necessarily be included in the irredundant subset of $tlcovs$. Moreover, minimum cardinality covers are necessarily irredundant, so we did not see any benefit to pruning by irredundancy in these experiments.

As a point of reference we include Table II, which lists performance metrics for past work using the MPC. However, a direct comparison is difficult since past work used different experimental setups and metrics. The accuracy metric reported by Raghavan and Mooney [39] gives partial credit for predictions when not all parameters are correct. The method of Tecuci and Porter [44] only predicts top-level task *schema*, not parameters. Likewise for the metric reported by Blaylock and Allen [6]; in cases where the schema were correct, on average only 41.4% of the parameters were also correctly identified. On the other hand, these metrics were all calculated with methods that make ≤ 4 predictions per example, so in many cases they are substantially more precise than our approach.

TABLE II: Performance Comparison, Monroe Plan Corpus

Method	Performance
This study	99.1%
Raghavan and Mooney [39]	98.9%
Tecuci and Porter [44]	99.8%
Blaylock and Allen [6]	97.4%

V. DISCUSSION

We have introduced, analyzed, and empirically evaluated a general-purpose, cognitive-level imitation learning framework, based on an extension of Parsimonious Covering Theory to support causal chaining and ordered effects. Causal inference of a demonstrator's intentions was found to enable rapid learning and generalization from a single demonstration. Our framework has been evaluated through imitation learning by a physical robot on a modest set of learning trials involving dock maintenance and block stacking. It was also evaluated through

[12]Given our theoretical results, if CAUSES is implemented correctly, then $tlcovs$ should really include the ground-truth 100% of the time. After manually inspecting a sample of failed examples, we believe the error is in fact often in the MPC itself, and not in our implementation of CAUSES. In some of the original plan trees, the parameters of some child tasks are inconsistent with their parents and would not accomplish the goal (see the Supplemental Online Information [42] for details). Unfortunately this data set is no longer supported, and since the repair of such errors by the authors of this paper might be viewed as introducing a post hoc bias, we simply note here that the 99.1% success rate should be viewed as a lower bound on true performance, which may even be somewhat better than we report.

a more extensive set of computer experiments using data from a third party (MPC). Experiments with human participants confirm that humans can also infer the intentions of a demonstrator and generalize to new situations on the basis of a single demonstration. This suggests that our use of causal inference leads to more human-like imitation abilities in a robot, and thus can enhance human-robot interaction. We conclude that using causal reasoning to construct parsimonious explanations is an effective approach to intention inference, and that using causal knowledge to infer a demonstrator's intentions, rather than trying to faithfully copy the demonstrator's actions, can be an effective approach to imitation learning.

It is worth noting that the first processing phase of EXPLAIN (Algorithm SSCOVERS, Appendix A) bears some resemblance to bottom-up parsing algorithms such as the CYK algorithm, in that it compactly represents multiple intersecting parse trees by storing their nodes in a dynamic programming table, where table entries are indexed by subsequences of the input [26]. However, there are also important differences. In SSCOVERS, table entries are also indexed by ℓ, the depth of the covering tree. In contrast, in the CYK algorithm, table entries are indexed by production rules in a finite context-free grammar. Production rules are analogous to elements $(u, \langle v \rangle)$ in the causal relation C, but since C may be infinite as it is in our case, the CYK approach cannot be applied. Even for fixed u, there may even be infinitely many $(u, \langle v \rangle) \in C$. For example, suppose both grippers are filled and the next intention is to toggle a switch. One of the gripped objects must be placed at a free spot on the dock. Since there are infinitely many free spots in continuous space, there are infinitely many $\langle v \rangle$ for the same u. Our approach works as long as the *converse* is false: there are only finitely many u for a given $\langle v \rangle$, as implied by our assumption that U is finite, and so the output of CAUSES($\langle v \rangle$) is finite. CAUSES can be viewed as the *inverse* of the "production rules" contained in C. In fact, a connection between PCT and parsing has been drawn before: Dasigi developed his own extension to PCT to account for causal chaining and ordered effects, in the context of natural language processing [12]. However, his technique came with no formal guarantees of correctness or efficiency.

As noted earlier, a number of researchers have explored various aspects of cognitive-level imitation learning, such as goal-directed imitation [17], [46], intention inference [25], [36], conceptual representations of sensory input [10], verbally-guided construction of HTN planning domains [30] and ontology-based object matching [7]. Researchers have also studied generalization from a single demonstration at the low level of motor trajectories [2], [50]. However, the details of these approaches are significantly different from ours, which we believe to be the only approach in which causal reasoning is used to interpret and generalize a single demonstration to new situations at a cognitive level. Designing an integrated framework that incorporates both our methods and those of other researchers could be a direction for future work. Future work should also evaluate our approach on larger sets of learning tasks, across a wider range of application scenarios, with other (potentially non-humanoid) robot embodiments, and with controlled end-user studies.

Our current formulation of PCT for imitation learning has some limitations that we hope to address in future work. First, it is questionable whether a tree structure is sufficiently expressive for all chained, ordered causal domains. One can imagine that in fortuitious circumstances, one child intention accomplishes two parent intentions, so the cover is not a tree, and our current formulation does not accommodate this possibility. For example, moving block A from B to C might accomplish two parent intentions: Getting A to C, and clearing off B. Second, it would be preferable to support *partial orderings* in addition to total orderings, especially in domains involving bimanual coordination like ours. Lastly, the original formulation of PCT included a probability calculus that models *causal probabilities*, which are distinct from and preferable to conditional probabilities in certain regards [38]. Incorporating causal probabilities into our approach would enable probability-based heuristics for guiding the covering process, perhaps leading to a more efficient algorithm.

Another shortcoming of our current implementation is simplistic sensorimotor processing that does not provide human-level motor control or visual feedback to the cognitive system. Future work will integrate our cognitive framework with neuro-inspired sensorimotor processing techniques that we have been developing separately [18], [34], [35], which will lead to better performance and eventually more natural human-robot interaction. It is worth pointing out that as our experiments show, people do not have perfect performance either, although we conjecture that this is for a different reason than the robot: the limitations of human short-term memory.

A third issue is the need for a domain author to expend significant effort encoding detailed background knowledge in the CAUSES function. One promising strategy to remedy this issue is to combine our work with complementary techniques in the literature that can learn causal relationships from experience and infer low-level intentions through mental simulation. For example, Oztop et al. formulated a biologically plausible model that could observe the beginning of an external agent's reaching motion, and infer a likely target position or grasp [36]. More recently, Ugur et al. modeled several stages of development in a physical robot that learned to emulate the low-level goals of a short demonstration, such as the final placement of an object after a single pick-up and put-down [46]. While these methods do not address high-level, cognitive imitation over multiple levels of causation and through multiple intentions over time, as EXPLAIN does, they do provide a means for a robot to *learn* a CAUSES function, in whole or in part, rather than relying solely on a domain author. In fact, since EXPLAIN treats CAUSES as a black box, the domain author could in principle implement CAUSES using a sub-routine that runs one of these biologically plausible models as part of its computation, assuming the model has already been trained.

An additional strategy for alleviating the burden on the domain author is to automatically incorporate newly learned skills back into the CAUSES function. For example, suppose CAUSES is implemented using schema templates as described in Sect. III-D, and suppose a sequence of top-level intentions $\langle (s_i, t_i, \langle x \rangle^{(i)}) \rangle_{i=1}^{m}$ has just been inferred from a new

demonstration. If the user provides a new name for the skill they just demonstrated, say $\bar{t}$, then a new template $(\bar{t}, \langle t_i \rangle)$ can be dynamically and automatically added to the list of templates in CAUSES. In the simplest case, default logic can be used to directly copy states and parameters from each t_i to the new $\bar{t}$, rather than relying on custom code as was done for the original built-in knowledge. The result is that in future demonstrations, $\bar{t}$ will be present as an even higher-level intention than was previously available, which can be hypothesized as one or more roots in the covering forests constructed by EXPLAIN. This process could be bootstrapped every time a new demonstration is provided. Supporting such an ability would allow demonstrators to substantially enrich the robot's causal knowledge over time.

APPENDIX A
PARSIMONIOUS COVERING ALGORITHMS

Given a causal inference problem, the solution (i.e., the set of all top-level covers) is computed by EXPLAIN (Fig. 5) in two phases. The first phase uses dynamic programming to compute every singleton cover of every contiguous subsequence of the observations. The second phase generates every top-level cover by carefully combining singleton covers from the first phase, pruning out the combinations that are not top-level. The first phase relies on an externally provided CAUSES function, which is EXPLAIN's interface to the causal knowledge base.

EXPLAIN combines the two phases to produce the final set of top-level covers for an observed sequence. The inputs to EXPLAIN are the function CAUSES, which encodes the problem domain, M, the domain constant defined in Sect. III-B2, and $\langle w_n \rangle_{n=1}^{N}$, the observed sequence to be explained. In the case of imitation learning, $\langle w_n \rangle_{n=1}^{N}$ is a demonstration consisting of N actions. M is provided explicitly because it may not be automatically computable from the CAUSES function alone. The output of EXPLAIN is $tlcovs$, the set of all top-level covers. $tlcovs$ can be pruned by additional parsimony criteria as a post-processing step. EXPLAIN invokes sub-routines SSCOVERS to perform the first phase and TLCOVERS to perform the second phase (described below). TLCOVERS is treated as an iterator so that, in rare cases when $|tlcovs|$ is very large, it can be terminated early or pruned by additional parsimony criteria online.

The algorithm SSCOVERS (Fig. 6) takes the same inputs as EXPLAIN: the CAUSES function (which encodes the domain), the bound M defined above, and the observed sequence $\langle w_n \rangle_{n=1}^{N}$. A dynamic programming table g is populated in an incremental, bottom-up fashion. The table entry $g_{j,k}^{\ell}$ accumulates the singleton ℓ-covers of $\langle w_n \rangle_{n=j+1}^{k}$ (i.e., those whose covering trees have depth at most ℓ). Along with each singleton cover u, its immediate children $\langle v \rangle$ in the covering tree are also stored for use in TLCOVERS (described later). Each outer iteration of SSCOVERS (lines 5-18) populates the ℓ-covers during the ℓ^{th} iteration, using the $(\ell-1)$-covers that were found in the previous iteration. Every $g_{j,k}^{0}$ is initialized with the trivial singleton covers, one for each w_k (lines 2-4). Line 6 initializes the ℓ-covers for the current iteration with those from the previous, since any $(\ell-1)$-cover is also an ℓ-cover.[13] Lines 7-13 check every subsequence $\langle u \rangle$ of every cover found so far to see if it has a higher-level cause $\tilde{u}$. Line 7 limits the search to $\langle u \rangle$ of length $m \leq M$, since by definition no sequence longer than M has any causes. Every such $\langle u \rangle$ will partition a subsequence of the original $\langle w_n \rangle_{n=1}^{N}$ into m disjoint, contiguous, consecutive subsequences, each of which are the leaves of the i^{th} covering tree rooted in u_i. That is, u_i covers $\langle w_n \rangle_{k_{i-1}+1}^{k_i}$, with $0 \leq k_0 < ... < k_m \leq N$. Line 8 enumerates every way $\langle w_n \rangle_{n=1}^{N}$ might be so partitioned. Each such $\langle u \rangle$ is then enumerated on line 9, based on the fact that if u_i has already been found to cover $\langle w_n \rangle_{k_{i-1}+1}^{k_i}$, then it will be contained[14] in $g_{k_{i-1},k_i}^{\ell-1}$. The set on this line draws each u_i from $g_{k_{i-1},k_i}^{\ell-1}$, which contains all singleton $(\ell-1)$-covers for the i^{th} subsequence of $\langle w_n \rangle_{n=1}^{N}$ in the current partition. If the current $\langle u \rangle$ has any causes, they each constitute a new ℓ-cover of $\langle w_n \rangle_{n=k_0+1}^{k_m}$ and are added to g_{k_0,k_m}^{ℓ} (line 10). When the current iteration of the outer loop has identified no new covers, the algorithm can be terminated. The current g^{ℓ} includes all singleton sub-covers from the previous iterations, so it is renamed g and returned (lines 14-17).

The second phase is performed by TLCOVERS (Fig. 7), which generates every top-level cover using the output g of SSCOVERS. Instead of a **return** statement, TLCOVERS uses **yield** statements[15] which save its state, suspend execution, and pass the current top-level cover to its "caller" (typically a for-loop, as in algorithm EXPLAIN). When the caller requests the next top-level cover, execution resumes where it left off and continues until **yield** is encountered again. The **yield** keyword makes TLCOVERS an iterable construct that a for-loop can invoke to receive parsimonious covers one at a time.

TLCOVERS is also recursive. It takes as input g (as computed by SSCOVERS), N (the length of the observed sequence), the bound M as defined above, and two "accumulators" $\langle u_i \rangle_{i=1}^{I}$ and $\langle k_i \rangle_{i=0}^{I}$: The current top-level cover is accumulated in $\langle u_i \rangle_{i=1}^{I}$, and the indices at which it partitions $\langle w_n \rangle_{n=1}^{N}$ are accumulated in $\langle k_i \rangle_{i=0}^{I}$. These are called "accumulators" because they accumulate a return value over the course of recursion. They are empty at the shallowest layer of the recursion, they contain a leading portion of a top-level cover mid-way down the recursion, and they contain a full top-level cover at the deepest layers of the recursion, at which point the full top-level cover is passed back up the recursive stack via the **yield** statements. The algorithm is initiated at the top-level of the recursion by calling TLCOVERS(g,N,M,$\langle\rangle$,$\langle 0 \rangle$) in a for-loop, as in EXPLAIN.

The algorithm starts by checking whether a cover for the full $\langle w_n \rangle_{n=1}^{N}$ has been accumulated, in which case the final partition point k_I will be N, the full sequence length (line 2). If so, it yields the accumulated cover (line 2). Otherwise, only a leading subsequence of $\langle w_n \rangle_{n=1}^{N}$ has been covered so far, and the algorithm enumerates all options for the next

[13] This is because, by definition, an $(\ell-1)$-cover has depth at most $(\ell-1)$. Therefore it has depth at most ℓ. So it is also an ℓ-cover. However the converse is not true.

[14] Technically, we cannot write $u_i \in g_{k_{i-1},k_i}^{\ell-1}$, since each cell in g actually contains pairs $(u, \langle v \rangle)$. At this stage, we only need one such pair containing u_i, hence the the $\langle v \rangle$ is existentially quantified in the set on line 9.

[15] The **yield** construct here is borrowed from Python.

procedure EXPLAIN(CAUSES,M,$\langle w_n \rangle_{n=1}^{N}$)
  $g \leftarrow$ SSCOVERS(CAUSES, M, $\langle w_n \rangle_{n=1}^{N}$) ▷ Phase 1: Generate all singleton sub-covers
  $tlcovs \leftarrow \{\}$
  for $t \in$ TLCOVERS(g, N, M,$\langle\rangle$,$\langle 0 \rangle$) **do** ▷ Phase 2: Generate all top-level covers
    $tlcovs \leftarrow tlcovs \cup \{t\}$
  end for
  return $tlcovs$
end procedure

Fig. 5: Explaining an observed sequence.

procedure SSCOVERS(CAUSES, M, $\langle w_n \rangle_{n=1}^{N}$)
  for $0 \leq j < k \leq N$ **do** ▷ Initialize g^0
    $g_{j,k}^{0} \leftarrow \{(w_k, \langle\rangle)\}$ **if** $j+1=k$ **else** $\emptyset$ **end if**
  end for
  for $\ell \in \langle 1, 2, ... \rangle$ **do** ▷ Populate g bottom-up
    $g_{j,k}^{\ell} \leftarrow g_{j,k}^{\ell-1}$ **for** $0 \leq j < k \leq N$ ▷ Initialize g^ℓ to $g^{\ell-1}$
    for $m \in \langle 1, 2, ..., M \rangle$ **do** ▷ Enumerate every $(\ell-1)$-cover of every subsequence of $\langle w \rangle$
      for $0 \leq k_0 < ... < k_m \leq N$ **do**
        for $\langle u \rangle \in \{\langle u_i \rangle_{i=1}^{m} \,|\, \forall i \, \exists \langle v \rangle \, (u_i, \langle v \rangle) \in g_{k_{i-1},k_i}^{\ell-1}\}$ **do**
          $g_{k_0,k_m}^{\ell} \leftarrow g_{k_0,k_m}^{\ell} \cup \{(\tilde{u}, \langle u \rangle) \,|\, \tilde{u} \in \text{CAUSES}(\langle u \rangle)\}$ ▷ Add new singleton sub-covers
        end for
      end for
    end for
    if $g_{j,k}^{\ell} = g_{j,k}^{\ell-1}$ **for** $0 \leq j < k \leq N$ **then** ▷ No new covers found, terminate
      $g \leftarrow \{g_{j,k}^{\ell} \,|\, 0 \leq j < k \leq N\}$
      return g
    end if
  end for
end procedure

Fig. 6: Singleton Sub-Cover Generation.

procedure TLCOVERS(g,N,M,$\langle u_i \rangle_{i=1}^{I}$,$\langle k_i \rangle_{i=0}^{I}$)
  if $k_I = N$ **then yield** ($\langle u_i \rangle_{i=1}^{I}$,$\langle k_i \rangle_{i=0}^{I}$) ▷ Accumulator covers full input sequence, yield it
  else ▷ Accumulator covers leading input subsequence, continue covering
    for $k_{I+1} \in \{k_I + 1, ..., N\}$ **do** ▷ Enumerate every possible partition point for next u_{I+1} to cover
      for $(u_{I+1}, \langle v \rangle) \in g_{k_I,k_{I+1}}$ **do**
        if $\exists m < M \; \exists \tilde{u} \, (\tilde{u}, \langle u_{I+1-m}, ..., u_{I+1} \rangle) \in g_{k_{I-m},k_{I+1}}$ **then** ▷ Skip current u_{I+1} if result is mid-level
          continue
        end if
        for $(\langle \hat{u} \rangle, \langle \hat{k} \rangle) \in$ TLCOVERS($g, N, M, \langle u_i \rangle_{i=1}^{I+1}, \langle k_i \rangle_{i=0}^{I+1}$) **do** ▷ Recursively cover rest of input sequence
          yield $(\langle \hat{u} \rangle, \langle \hat{k} \rangle)$
        end for
      end for
    end for
  end if
end procedure

Fig. 7: Top-level Cover Generation.

partition point in the tail (line 4). For each option k_{I+1}, it enumerates all singleton covers u_{I+1} that could be appended to the cover accumulated so far (line 5). For each singleton, it checks whether appending would result in a cover that is mid-level, with some higher-level cause $\tilde{u}$ (line 6). If so, the current singleton is skipped (line 7). Otherwise, it is added to the accumulator, and the algorithm is called recursively (line 9) to finish accumulating. Each top-level cover that results is yielded to the caller one by one (line 10).

The first iterates yielded by TLCOVERS are those where the leading covering trees have the fewest leaves and the trailing covering trees have the most. This is an arbitrary byproduct of the loop order on line 4, since the next k_{I+1} is searched from head to tail. If ordering were important, it could be modified, although we did not explore it further in this work. For example, if k_{I+1} started at $\lfloor (k_I + N)/2 \rfloor$ and worked outwards, the first iterates of TLCOVERS would tend to have leaves more uniformly distributed among the covering trees.

APPENDIX B
THEORETICAL RESULTS

Here we prove that Algorithm EXPLAIN is correct and efficient. N, M, U, and L are as defined in Sect. III-B1.

Theorem 1 shows that SSCOVERS is correct: after the algorithm completes, each entry in the dynamic programming table $g^{\ell}_{j,k}$ contains all and only the singleton ℓ-covers of $\langle w_n \rangle_{n=j+1}^{k}$.

Theorem 1. *Let g be the return value of* SSCOVERS$(\text{CAUSES}, M, \langle w_n \rangle_{n=1}^{N})$, *and let $\langle w_n \rangle_{n=j+1}^{k}$ be any subsequence of $\langle w_n \rangle_{n=1}^{N}$. For every $(u, \langle v \rangle) \in g_{j,k}$, the singleton $\langle u \rangle$ covers $\langle w_n \rangle_{n=j+1}^{k}$* (soundness). *For every singleton cover $\langle u \rangle$ of $\langle w_n \rangle_{n=j+1}^{k}$, $(u, \langle v \rangle) \in g_{j,k}$ for some $\langle v \rangle$* (completeness).

Proof. The proof is by induction on ℓ. After line 4, each $g^{0}_{j,k}$ contains all and only the 0-covers of $\langle w_n \rangle_{n=j+1}^{k}$, namely, the trivial covers where each w_k covers itself. Now assume the inductive hypothesis: on the ℓ^{th} iteration of lines 5-18, every $g^{\ell-1}_{j,k}$ contains all and only the $(\ell-1)$-covers of $\langle w_n \rangle_{n=j+1}^{k}$.

For soundness, consider $(u, \langle v \rangle) \in g^{\ell}_{j,k}$. If it was acquired via line 6, it was already in $g^{\ell-1}_{j,k}$ and covers $\langle w_n \rangle_{n=j+1}^{k}$ by the inductive hypothesis. Otherwise, it was added on line 10 when $k_0 = j$ and $k_m = k$, in which case $(u, \langle v \rangle)$ is in C by the definition of CAUSES. Moreover, $\langle v \rangle$ is in the set on line 9. Therefore each v_i is stored in some $g^{\ell-1}_{k_{i-1},k_i}$, and is a singleton cover by the inductive hypothesis. So all parent-child relationships in the sub-trees rooted at each v_i are in C as well. Therefore the tree formed by making $\langle v \rangle$ the ordered children of u is also a covering tree, and consequently u is a singleton cover of $\langle w_n \rangle_{n=j+1}^{k}$.

For completeness, suppose that $\tilde{u}$ is a singleton ℓ-cover of some subsequence $\langle w_n \rangle_{n=j+1}^{k}$. Fix an associated covering tree and let $\langle u_i \rangle_{i=1}^{\tilde{m}}$ be the ordered children of $\tilde{u}$ in the tree. By definition, each u_i is the root of a sub-tree with depth at most $\ell - 1$, and $\langle w_n \rangle_{n=j}^{k}$ is the concatenation of the ordered leaves of each sub-tree. Let $\tilde{k}_{i-1}+1$ and $\tilde{k}_i$ be the starting and ending indices of the leaves of the i^{th} sub-tree. By the inductive hypothesis, each u_i is stored in $g^{\ell-1}_{k_{i-1},k_i}$. Therefore $\langle u_i \rangle_{i=1}^{\tilde{m}}$ will be included in the set on line 9 when $m = \tilde{m}$ and each $k_i = \tilde{k}_i$. Since $(\tilde{u}, \langle u_i \rangle_{i=1}^{\tilde{m}})$ is a parent-child relationship in the covering tree, it is also in C, and by the definition of CAUSES, it will be added to $g^{\ell}_{k_0,k_m} = g^{\ell}_{j,k}$ on line 10. □

To prove that SSCOVERS is computationally efficient, we first establish a lemma that bounds the number of singleton covers in each $g^{\ell}_{j,k}$.

Lemma 1. *The cardinality of any $g^{\ell}_{j,k}$ is at most $\mathcal{O}(MU^{M+1}N^{M-1})$.*

Proof. There are $\sum_{m=1}^{M} \binom{k-j-1}{m-1}$ partitions of the form $j = k_0 < ... < k_m = k$ splitting $\langle w_n \rangle_{n=j+1}^{k}$ into m subsequences. $\sum_{m=1}^{M} \binom{k-j-1}{m-1}$ is $\mathcal{O}(\sum_{m=1}^{M} \binom{N-1}{m-1})$, which simplifies[16] to $\mathcal{O}(MN^{M-1})$. For every such partition, each of the m subsequences has at most U singleton covers, so there are at most U^m associated covers. Therefore there are $\mathcal{O}(MU^{M}N^{M-1})$ possible covers $\langle v \rangle$ of $\langle w_n \rangle_{n=j+1}^{k}$. Each element of $g^{\ell}_{j,k}$ pairs one of at most U singleton covers u with one of these $\langle v \rangle$. Therefore $|g^{\ell}_{j,k}|$ is $\mathcal{O}(MU^{M+1}N^{M-1})$. □

The upper bound in Lemma 1 is rather loose. It counts every possible pairing of u's with $\langle v \rangle$'s, ignoring whether each $(u, \langle v \rangle)$ is actually in C, resulting in a significant overestimate. Equipped with Lemma 1, we can prove that SSCOVERS completes in polynomial time.

Theorem 2. *The worst case complexity of* SSCOVERS$(\text{CAUSES}, M, \langle w_n \rangle_{n=1}^{N})$ *is polynomial in N.*

Proof. Lines 2-4 take $\mathcal{O}(N^2)$ steps since j and k range from 0 to N. Line 10 takes time constant in N to compute CAUSES. The cardinality of the resulting set is $\mathcal{O}(U)$ so the union operation takes time $\mathcal{O}(U)$ using an efficient (e.g., hash-based) set implementation for each $g^{\ell}_{j,k}$. The cardinality of the set on line 9 is $\mathcal{O}((MU^{M+1}N^{M-1})^{M}) = \mathcal{O}(M^{M}U^{M^2+M}N^{M^2-M})$, since each u_i is chosen from one of at most M sets $g^{\ell-1}_{k_{i-1},k_i}$, and each $g^{\ell-1}_{k_{i-1},k_i}$ contains $\mathcal{O}(MU^{M+1}N^{M-1})$ options for u_i by Lemma 1. Line 8 iterates over $\binom{N+1}{M+1}$ partitions which simplifies to $\mathcal{O}(N^{M+1})$. Line 6 copies $\mathcal{O}(N^2)$ sets each of size at most $\mathcal{O}(MU^{M+1}N^{M-1})$, again by Lemma 1. Therefore the total complexity of lines 6-17 is

$$\mathcal{O}(MU^{M+1}N^{M+1} + M^{M+1}U^{M^2+M}N^{M^2+1}(U+X)),$$

where X is the complexity of CAUSES. Although line 5 has no termination condition, all singleton covers have depth at most L, where L is the bound on causal chain depth defined previously, assumed to be finite.[17] Since correctness was shown in Theorem 1 by induction on ℓ, the termination check on line 14 will be satisfied after at most L iterations. Therefore the total complexity of the algorithm is

$$\mathcal{O}(L(MU^{M+1}N^{M+3} + M^{M+1}U^{M^2+M}N^{M^2+1}(U+X))).$$

[16] The term $\binom{A}{B}$ is $\mathcal{O}(A^B)$ since it has a numerator with B factors, each at most A.

[17] In fact, our results still hold if the supremum L is infinite, as long as every causal chain and every demonstration is finite. If every causal chain is finite, then every covering forest of a finite demonstration has finite depth and our algorithms will terminate.

□

The exponential dependence on M^2 is a theoretical concern but we do not consider it a serious defect in practice. The main reason is that our empirical run times are very reasonable even on fairly long demonstrations (see Sect. IV). In addition, we believe that in practice M is typically rather small. It is at most 6 in our own robotics domain and the unrelated Monroe Plan Corpus, and most child sequences have length closer to 2 or 3.

Next, we prove that the second phase of causal inference is also correct. Note that top-level-ness is our primary parsimony criteria, and implies irredundancy except in rare pathological cases. As such, Theorem 3 shows that TLCOVERS yields all and only the parsimonious covers of an observed sequence.

Theorem 3. *Let g be the return value of* SSCOVERS$(\text{CAUSES}, M, \langle w_n \rangle_{n=1}^{N})$. *Iteration over* TLCOVERS$(g, N, M, \langle\rangle, \langle 0 \rangle)$ *yields all* (completeness) *and only* (soundness) *the top-level covers of $\langle w_n \rangle_{n=1}^{N}$.*

Proof. For completeness, suppose $\langle u_i \rangle_{i=1}^{\tilde{I}}$ is a top-level cover of $\langle w_n \rangle_{n=1}^{N}$. We can show that $\langle u_i \rangle_{i=1}^{\tilde{I}}$ is yielded by a reverse induction on I ranging from $\tilde{I}$ to 0. Let $0 = \tilde{k}_0 < ... < \tilde{k}_{\tilde{I}} = N$ be the partition of $\langle w_n \rangle_{n=1}^{N}$ induced by $\langle u_i \rangle_{i=1}^{\tilde{I}}$, i.e., u_i covers $\langle w_n \rangle_{n=\tilde{k}_{i-1}+1}^{\tilde{k}_i}$. For the base case, when $I = \tilde{I}$, the call TLCOVERS$(g, N, M, \langle u_i \rangle_{i=1}^{I}, \langle \tilde{k}_i \rangle_{i=1}^{I})$ will execute line 2 since $\tilde{k}_{\tilde{I}} = N$. For the inductive case, suppose that TLCOVERS$(g, N, M, \langle u_i \rangle_{i=1}^{I+1}, \langle \tilde{k}_i \rangle_{i=1}^{I+1})$ yields $\langle u_i \rangle_{i=1}^{\tilde{I}}$; we must show the same for TLCOVERS$(g, N, M, \langle u_i \rangle_{i=1}^{I}, \langle \tilde{k}_i \rangle_{i=1}^{I})$. During this call, the iteration on line 4 will eventually reach $k_{I+1} = \tilde{k}_{I+1}$ since $\tilde{k}_{I+1} \in \{\tilde{k}_I + 1, ..., N\}$. Since u_{I+1} is a singleton cover of $\langle w_n \rangle_{n=\tilde{k}_I+1}^{\tilde{k}_{I+1}}$, it is stored in $g_{\tilde{k}_I+1, \tilde{k}_{I+1}}$ and will be included in the iteration on line 5. Since $\langle u_i \rangle_{i=1}^{\tilde{I}}$ is top-level, the check on line 6 will fail. Therefore the iteration on line 9 will be reached, and by the inductive hypothesis, it will yield $\langle u_i \rangle_{i=1}^{\tilde{I}}$, which gets passed up the recursion. Running the induction through to $I = 0$, we have that TLCOVERS$(g, N, M, \langle\rangle, \langle 0 \rangle)$ yields $\langle u_i \rangle_{i=1}^{\tilde{I}}$.

For soundness, let $\langle u_i \rangle_{i=1}^{\tilde{I}}$ be any iterate yielded by TLCOVERS$(g, N, M, \langle\rangle, \langle 0 \rangle)$. We must show that it is a top-level cover. Each u_i was accumulated at some recursion depth as some iterate u_{I+1} in line 5. Since u_{I+1} is drawn from $g_{k_I, k_{I+1}}$, it is a singleton cover of $\langle w_n \rangle_{n=k_I+1}^{k_{I+1}}$, and therefore $\langle u_i \rangle_{i=1}^{\tilde{I}}$ is a cover for the full $\langle w_n \rangle_{n=1}^{N}$. If it were mid-level and not top-level, there would be some subsequence $\langle u_i \rangle_{i=I-m}^{I+1} \sqsubseteq \langle u_i \rangle_{i=1}^{\tilde{I}}$ for some $m < M$ and $I < \tilde{I}$ that could be covered by some higher-level cause $\tilde{u}$. But then the check on line 6 would be true at recursion depth I, so $\langle u_i \rangle_{i=1}^{I+1}$ would have never been passed to the recursive call on line 9, and $\langle u_i \rangle_{i=1}^{\tilde{I}}$ would have never been yielded. Therefore $\langle u_i \rangle_{i=1}^{\tilde{I}}$ must be top-level. □

Having shown correctness, it remains to characterize the efficiency of TLCOVERS.

Theorem 4. *The worst-case complexity of* TLCOVERS$(g, N, M, \langle\rangle, \langle 0 \rangle)$ *is polynomial in N and T, where T is the number of top-level covers.*

Proof. The recursive execution trace of TLCOVERS$(g, N, M, \langle\rangle, \langle 0 \rangle)$ can be viewed as a tree, where each node corresponds to some $\langle u_i \rangle_{i=1}^{I+1}$ that gets checked on line 6, and if it passes the check, gets passed to the recursive call on line 9. Note there is no connection between this recursion tree and the notion of a covering tree. The depth in the tree is equal to the depth in the recursion. The node for any $\langle u_i \rangle_{i=1}^{I+1}$ is a child of the node for $\langle u_i \rangle_{i=1}^{I}$, and the root is associated with $\langle\rangle$. Line 6 compares a length-$\mathcal{O}(M)$ sequence against each of $\mathcal{O}(MU^{M+1}N^{M-1})$ elements in each of M sets $g_{k_{I-m}, k_{I+1}}$ by Lemma 1, so the worst-case run time of the algorithm is proportional to the size of the tree times $M^3U^{M+1}N^{M-1}$.

The leaves of the tree can be split into two groups. The "good" leaves are top-level covers of the full $\langle w_n \rangle_{n=1}^{N}$, which get yielded and passed up the recursion on line 2. The "bad" leaves are mid-level covers of leading subsequences of $\langle w_n \rangle_{n=1}^{N}$, which fail the check on line 6 and prevent a recursive call. Likewise, the nodes of the tree can be split into two groups: the "good" nodes are those from which good leaves are reachable, and the "bad" nodes are the rest. It follows that all descendents of a bad node are also bad.

There are T good leaves, and each is reachable from at most N nodes along the path from the root, since the length of a cover is never more than the length of the covered sequence. So there are $\mathcal{O}(TN)$ good nodes. Lines 4 and 5 enumerate $\mathcal{O}(MU^{M+1}N^{M})$ iterates by Lemma 1, so each good node has $\mathcal{O}(MU^{M+1}N^{M})$ bad children, and each bad child is the root of a sub-tree containing only bad nodes. Moreover, every bad node is in some such sub-tree. The depth of these sub-trees is at most M, since line 6 identifies any mid-level cover within M recursive steps. The branching factor of the sub-trees is again $\mathcal{O}(MU^{M+1}N^{M})$, so the size of the sub-tree is $\mathcal{O}(M^{M}U^{M^2+M}N^{M^2})$. Therefore there are at most $\mathcal{O}(TM^{M}U^{M^2+M}N^{M^2})$ bad nodes. The total size of the entire tree is the sum of good and bad node counts, so the total complexity of the algorithm is

$$\mathcal{O}(M^3U^{M+1}N^{M-1}(TN + TM^{M}U^{M^2+M}N^{M^2})).$$ □

Another theoretical concern is that the number of top-level covers T is not independent from the length of the demonstration sequence N, and may have exponential dependence on N, thereby concealing a worse-than-polynomial run time. Indeed, our empirical results do show that in complex domains, T can be very large in some cases (Sect. IV-B). However, T was rarely large enough that EXPLAIN ran for an impractical amount of time, and the size of the output $|tlcovs|$ could be mitigated by pruning with additional parsimony criteria.

ACKNOWLEDGEMENTS

Supported by ONR award N000141310597. Thanks to Ethan Reggia for building the hard drive docking station.

REFERENCES

[1] Akgun, B., Cakmak, M., Yoo, J. W., Thomaz, A. L: Trajectories and Keyframes for Kinesthetic Teaching. In: *Proc. of the* 7^{th} *Annu. ACM/IEEE Intl. Conf. on Human-Robot Interaction*, pp. 391–398. ACM (2012)

[2] Atkeson, C.G., Schaal, S.: Learning Tasks from a Single Demonstration. In: *Proc. of the 1997 IEEE Intl. Conf. on Robotics and Automation* Vol. 2, pp. 1706-1712. IEEE (1997)

[3] Baldwin, D., Baird, J.: Discerning Intentions in Dynamic Human Action. *Trends in Cog. Sciences* 5(4), pp. 171–178 (2001)

[4] Barros, J., Serra, F., Santos, V., Silva, F.: Tele-Kinesthetic Teaching of Motion Skills to Humanoid Robots through Haptic Feedback. In: *IEEERAS Humanoids' 2014 Workshop on Policy Representations for Humanoid Robots* (2014)

[5] Blaylock, N., Allen, J.: Generating Artificial Corpora for Plan Recognition. In: *Ardissono, L., Brna, P., Mitrovic, A. (eds.) User Modeling 2005* LNAI, vol. 3538, pp. 179-188. Springer, Edinburgh (2005)

[6] Blaylock, N., Allen, J.: Hierarchical Instantiated Goal Recognition. In: *AAAI Workshop on Modeling Others from Observations* (2006).

[7] Boteanu, A., Kent, D., Mohseni-Kabir, A., Rich, C., Chernova, S.: Towards Robot Adaptability in New Situations. In: *AAAI Fall Symposium Series* (2015)

[8] Carberry, S.: Techniques for Plan Recognition. *User Modeling and User-Adapted Interaction* 11(1-2), pp. 31–48 (2001)

[9] Chater, N., Vitányi, P.: Simplicity: A Unifying Principle in Cognitive Science? *Trends in Cog. Sciences* 7(1), pp. 19–22 (2003)

[10] Chella, A., Dindo, H., Infantino, I.: A Cognitive Framework for Imitation Learning. *Robotics and Autonomous Systems* 54(5), pp. 403–408 (2006)

[11] Dasigi, V., Reggia, J.: Parsimonious Covering as a Method for Natural Language Interfaces to Expert Systems. *AI in Medicine* 1(1), pp. 49–60, (1989)

[12] Dasigi, V.: Parsing = Parsimonious Covering?: Abduction in Logical Form Generation. In: *Proc. of the 12th Intl. Joint Conf. on A. I.* Vol. 2. Morgan Kaufmann Publishers Inc. (1991)

[13] Dindo, H., Chella, A., La Tona, G., Vitali, M., Nivel, E., Thórisson, K. R.: Learning Problem Solving Skills from Demonstration. In: *Schmidhuber, J., Thórisson, K. R., Looks, M. (eds.) AGI 2011* LNCS, vol. 6830, pp. 194–203. Springer Berlin Heidelberg (2011)

[14] Fikes, R. E., Nilsson, N. J.: STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence* 2(3-4), pp. 189-208 (1971)

[15] Fitzgerald, T., Goel, A. K., Thomaz, A. L.: Representing Skill Demonstrations for Adaptation and Transfer. In: *AAAI Symposium on Knowledge, Skill, and Behavior Transfer* (2014)

[16] Fogassi, L., Ferrari, P.F., Gesierich, B., Rozzi, S., Chersi, F., Rizzolatti, G.: Parietal Lobe: from Action Organization to Intention Understanding. *Science* 308, pp. 662667 (2005)

[17] Friesen, A. L., Rao, R. P.: Imitation Learning with Hierarchical Actions. In: *Proc. of the* 9^{th} *Intl. Conf. on Development and Learning*, pp. 263–268. IEEE (2010)

[18] Gentili, R. J., Oh, H., Huang, D.-W., Katz, G. E., Miller, R. H., Reggia, J. A.: Towards a Multi-Level Neural Architecture that Unifies Self-Intended and Imitated Arm Reaching Performance. In: *Proc. of the* 36^{th} *Annu. Intl. Conf. of the IEEE Engineering in Medicine and Biol. Society*, pp. 2537-2540. IEEE (2014)

[19] Ghallab, M., Nau, D., Traverso, P.: Automated Planning. Elsevier (2004)

[20] Haikonen, P.: The Cognitive Approach to Conscious Machines. Imprint Academic (2003)

[21] Henson, C., Sheth, A., Thirunarayan, K.: Semantic perception: Converting Sensory Observations to Abstractions. *Internet Computing*, 16(2), pp. 26–34. IEEE (2012)

[22] Huang, D.-W., Katz, G., Langsfeld, J. D., Oh, H., Gentili, R. J., Reggia, J. A.: An Object-Centric Paradigm for Robot Programming by Demonstration. In: *Schmorrow, D. D., Fidopiastis, M. C. (eds.) Foundations of Augmented Cognition 2015*. LNCS, vol. 9183, pp. 745–756. Springer Intl. Publishing (2015)

[23] Huang, D.-W., Katz, G., Langsfeld, J. D., Gentili, R. J., Reggia, J. A.: A Virtual Demonstrator Environment for Robot Imitation Learning. In: *IEEE Intl. Conf. on Technologies for Practical Robot Applications (TePRA)*, pp. 1-6. IEEE (2015)

[24] Iacoboni, M., Molnar-Szakacs, I., Gallese, V., Buccino, G., Mazziotta, J.C., Rizzolatti, G.: Grasping the Intentions of Others with Ones Own Mirror Neuron System. *PLoS Biol.* 3, e79 (2005)

[25] Jansen, B., Belpaeme, T.: A Computational Model of Intention Reading in Imitation. *Robotics and Autonomous Systems* 54(5), pp. 394–402. (2006)

[26] Kasami T. An Efficient Recognition and Syntax Analysis Algorithm For Context-Free Languages. Hawaii University Honolulu Dept. of Electrical Engineering Tech. Report (1965)

[27] Katz, G., Huang, D.-W., Gentili, R., Reggia, J.: Imitation Learning as Cause-Effect Reasoning. In: 9^{th} *Conf. on Artificial General Intelligence*. Springer Intl. Publishing. (2016)

[28] Kautz, H., Allen, J.: Generalized Plan Recognition. In: *AAAI* 86(3237), p. 5. (1986)

[29] Meltzoff, A., Moore, A.: Imitation of Facial and Manual Gestures by Human Neonates. *Science* 198(4312), pp. 75-78 (1977)

[30] Mohseni-Kabir, A., Rich, C., Chernova, S., Sidner, C., and Miller, D.: Interactive Hierarchical Task Learning from a Single Demonstration. In: *Proc. of the* 10^{th} *Annu. ACM/IEEE Intl. Conf. on Human-Robot Interaction*, pp. 205-212. ACM (2015)

[31] Nau, D.S., Au, T.C., Ilghami, O., Kuter, U., Murdock, J.W., Wu, D., Yaman, F.: SHOP2: An HTN planning system. *Journal of A. I. Research* 20, pp.379-404 (2003)

[32] Nau, D. S., Ghallab, M., Traverso, P.: Blended Planning and Acting. In: *Proc. of the* 29^{th} *AAAI Conf. on A. I.*, AAAI (2015)

[33] Nebel, B., Koehler, J.: Plan Reuse Versus Plan Generation. *Artificial Intelligence* 76(1), pp. 427–454 (1995)

[34] Oh, H.: A Multiple Representations Model of the Human Mirror Neuron System for Learned Action Imitation (Doctoral dissertation). (2015)

[35] Oh, H., Gentili, R. J., Reggia, J. A., Contreras-Vidal, J. L.: Modeling of Visuospatial Perspectives Processing and Modulation of the Fronto-Parietal Network Activity During Action Imitation. In: *2012 Annu. Intl. Conf. of the IEEE Engineering in Medicine and Biol. Society* pp. 2551-2554. IEEE (2012)

[36] Oztop, E., Wolpert, D., Kawato, M.: Mental state inference using visual control parameters. *Cog. Brain Research*, 22(2), pp. 129-151 (2005)

[37] Peng, Y., Reggia, J.: Diagnostic problemsolving with causal chaining. *Intl. Journal of Intelligent Systems* 2(3), pp. 265-302 (1987)

[38] Peng, Y., Reggia, J.: Abductive Inference Models for Diagnostic Problem-Solving. Springer-Verlag New York (1990)

[39] Raghavan, S., Mooney, R. J.: Bayesian Abductive Logic Programs. In: *Proc. of the* 10^{th} *AAAI Workshop on Statistical Relational A. I.* pp. 82-87 (2010)

[40] Rizzolatti, G., Craighero, L.: The Mirror-Neuron System. *Annu. Rev. Neurosci.* 27, pp. 169–192 (2004)

[41] Shivashankar, V., Alford, R., Kuter, U., Nau, D.: The GoDeL Planning System: A More Perfect Union of Domain-Independent and Hierarchical Planning. In: *Proc. of the* 23^{rd} *Intl. Joint Conf. on A.I.* AAAI (2013)

[42] Supplemental Online Information for this paper. http://www.cs.umd.edu/~reggia/supplement/index.html

[43] Sweeney, J. Grupen, R.: A Model of Shared Grasp Affordances from Demonstration. In: *7th Intl. Conf. on Humanoid Robots*, pp. 27–35. IEEE (2007)

[44] Tecuci, D., Porter, B.: Memory Based Goal Schema Recognition. In: *Proc. of the* 22^{nd} *Florida A.I. Research Society Conf.* (2009)

[45] Tuhrim, S., Reggia, J., Goodall, S.: An Experimental Study of Criteria for Hypothesis Plausibility. *Journal of Experimental and Theoretical A.I.* 3, pp. 129-144 (1991)

[46] Ugur, E., Nagai, Y., Sahin, E., Oztop, E.: Staged development of robot skills: Behavior formation, affordance learning and imitation with motionese. *IEEE Transactions on Autonomous Mental Development*, 7(2), pp. 119-139 (2015).

[47] Verma, D., Rao, R.: Goal-Based Imitation as Probabilistic Inference over Graphical Models. In: *Weiss, Y., Schölkopf, B., Platt, J. C. (eds.) Advances in Neural Information Processing Systems* 18, pp. 1393-1400. MIT Press (2006)

[48] de Melo Rezende, A., Wainer, J.: A temporal extension to the Parsimonious Covering Theory. *Advances in A.I.*, pp. 201-210. Springer Berlin Heidelberg (1996)

[49] Wen, F., Chang, C.: A New Approach to Fault Diagnosis in Electrical Distribution Networks Using a Genetic Algorithm. *A.I. in Engineering* 12(1), pp. 69–80 (1998)

[50] Wu, Y., Su, Y., Demiris, Y.: A Morphable Template Framework for Robot Learning by Demonstration. *Robotics and Autonomous Systems* 62(10), pp. 1517–1530 (2014)