Towards Explainable Tool Creation by a Robot

Handy Wicaksono[1,2] **Claude Sammut**[1] **Raymond Sheh**[3]
[1] School of Computer Science and Engineering, University of New South Wales; Sydney, Australia
[2] Department of Electrical Engineering, Petra Christian University; Surabaya, Indonesia
[3] Department of Computing, Curtin University; Perth, Australia
handyw@cse.unsw.edu.au, claude@cse.unsw.edu.au, Raymond.Sheh@curtin.edu.au

Abstract

A robot may benefit from being able to use a tool to solve a complex task. When an appropriate tool is not available, a useful ability for a robot would be to create a novel one based on its experiences. With the advent of inexpensive 3D printing, it is now possible to give robots such an ability. We propose CREATIVE, a relational approach to enable a robot to learn how to use an object as a tool and, if needed, to design and construct a new tool. The advantage of a relational approach is its interpretable learning results. To get meaningful explanations, we store the relevant knowledge during experiments, reason about them, and present them in a meaningful way to a human. Furthermore, a human user should be able to take action based on explanations given by a robot. We outline our plan to add this feature in CREATIVE and perform preliminary experiments on it.

1 Introduction

Humans use tools to solves complex problems in their daily life. When an appropriate tool is not available, we can innovate and design a new tool, often based on prior experience, or even invent it from scratch. Such abilities are also beneficial for a robot. Although there is increasing interest in tool use learning by a robot, there has been little work on tool creation. For example, Wang [2014] developed a robot that can create tools on the fly, but the design of the tools are predefined, and no learning is performed. Most of the existing work uses feature-based representations which are not expressive enough to be extended to tool creation.

Creating a novel tool by modifying or extending a previous design is called tool innovation in cognitive science [Beck *et al.*, 2011]. We take this approach as a means of limiting the search space of possible tools that the robot may try to design. The system uses a specific-to-general search, starting with a design of a known tool, but which either is not present or whose properties only partially match the requirements of the task. These properties are then modified by a generalization of the tool description.

We introduce a system called CREATIVE (Cognitive Robot Equipped with Autonomous Tool Invention Expertise), which refers to a robot that can learn how to use a tool and, if needed, design and build a new one. We use a relational representation of tool models, which are learned by a form of Inductive Logic Programming (ILP)[Sammut, 1981; Muggleton, 1991].

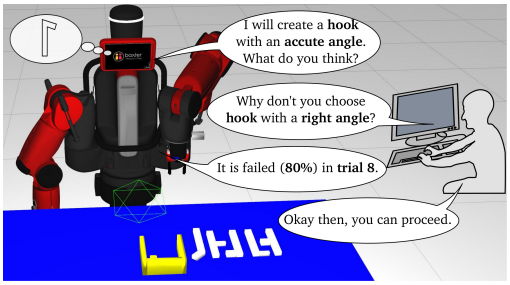

Figure 1: Baxter robot have a discussion with human about a novel tool it plans to create

Having an autonomous agent seems ruling out human user from the loop. However, recently there is an increasing demand to put human back in the loop, especially in critical fields, such as rescue, health, or military. Human user needs to know why (or why not) an AI agent makes such decisions, so he can trust it. We give its illustration in Fig. 1.

Explainable learning results is hard to achieve in "black box" AI mechanisms. Fortunately, CREATIVE utilises relational representation, so its results are inherently explainable as it describes the relation between objects as Prolog facts. Furthermore, it can give deeper explanation about robot's decisions compared than the feature-based one.

To summarize, here are the features that we propose to add in CREATIVE:

- Ability to debug the learned robot's plan so, if needed, errors can be corrected by human
- Ability to explain to human about why (or why not) the robot makes a decision
- Facility for human user to take action based on that explanations

In the following sections, we review relevant related work, describe our representation of states and actions, explain our learning mechanism, and present the preliminary experimental results.

2 Related Work

Previous work on tool use learning by a robot includes Stoytchev [2005], who demonstrated learning the tool affordances that are grounded in the robot's behaviors. Recent work by Tikhanoff et al. [2013] integrates exploratory behaviors and geometrical feature extraction to learn affordances and tool use. Mar et al. [2015] extend their work by learning the grasp configurations that affect the outcome of a tool use action. All of these systems use feature vector representations, and therefore, are limited in their ability to describe relations between components of a tool and their relations to other objects. A relational approach overcomes this limitation [Brown and Sammut, 2012].

There has been little prior work on tool creation. Wang et. al. [2014] developed a manipulator robot that can manufacture tools on the fly. However, the designs of tools are predefined, not learned. Brodbeck et al. [2015] performed evolutionary stochastic optimization of mechanical designs to construct a complex agent. This evolutionary approach is also limited to feature-based representations and learning takes a long time as it can not incorporate any background knowledge. ILP is capable of learning more expressive relational representation, and it can easily make use of background knowledge.

A relational representation also produces easy-to-interpret learning results, which are desirable in AI application. Recently there are increasing interests in this field, namely eXplainable Artificial Intelligence or XAI [DARPA, 2016], as many popular and accurate AI algorithms behave like black boxes. To tackle this issue in robotics, a verbalization is done so a mobile robot can "tell" its experience in a way that understandable by humans [Rosenthal *et al.*, 2016]. Other work enables a rescue robot to explain its actions to humans by converting a decision tree into a human-friendly information [Sheh, 2017]. Our representation is more general than previous work so that a deeper explanation can be expected.

3 States and Actions Representation

We maintain two levels of state representation: abstract and primitive. Primitive states contain quantitative values, such as the pose of objects in the world, while abstract states capture qualitative relationships between objects. As we operate in an ILP framework, state representations are expressed as a set of Prolog facts.

We define a tool as an object that is deliberately employed by an agent to help it achieve a goal that would otherwise be difficult or impossible to achieve. A tool possesses spatial and structural properties. We build a simple ontology of tools, where a general tool can be specified into a hook, a wedge, a hammer or other kind of tools. Each tool has unique structural and material properties. A hierarchy example of hook's properties is shown in Fig. 2.

We use a STRIPS [Fikes and Nilsson, 1971] action model to describe tool actions:

PRE: condition that must hold so that the action can be performed

EFFECTS: conditions that become true or false as a result of performing the action

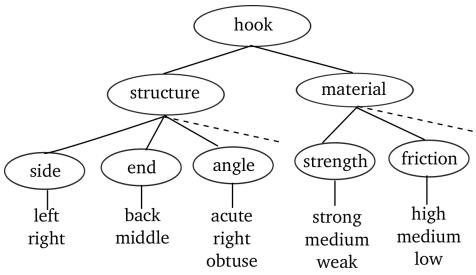

Figure 2: Properties of a hook

The action model does not provide any quantitative information needed to execute a motor command. For example, an action model might state that a hook must be placed behind a target object to be pulled, but it does not give a precise position. However, spatial literals in the action model's effects (e.g. behind(X, Y)) can be treated as constraints to a constraint solver [Apt *et al.*, 2007], which later produces a set of quantitive parameter values, any of which can be used to achieve the effects.

4 CREATIVE

We illustrate our learning framework for CREATIVE in Fig. 3. Initially, a robot does not have a complete action models, so it can not construct a plan. The robot can **learn by observing** a single correct example given by a tutor and build an initial novel model to complete the previous ones. A problem solver must achieve a goal and may use a tool to do it. Results of its attempt to solve the problem are sent to a critic that determines if they correspond to the expectations of a planner. Depending on that assessment, **learning by trial and error** is performed, via an ILP learner, to update a relevant action model. A problem generator selects a new experiment to test the updated model. If there are no suitable tool to accomplish the task, **tool invention** is performed. A label from a critic is passed to the tool generalizer and manufacturer. A simple **user interface** is added so a human user can get explanations from a robot.

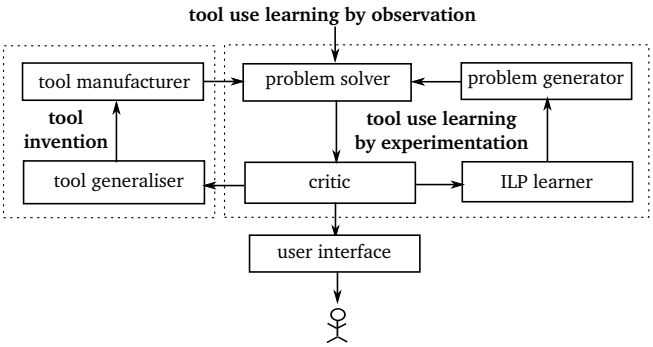

Figure 3: CREATIVE learning framework with a user interface

4.1 Tool Use Learning

In learning by observation stage, the robot is shown a correct example by a trainer, which is used to construct the initial action model. Primitive states are acquired by a vision sensor and grouped as segments. They are matched with existing action models. If there are segments that have no match, a new action models is constructed. Preconditions of this new action model contain a hypothesis which describes tool's required properties.

Our hypothesis formation adapts Mitchell's [Mitchell, 1977] version space approach in that the initial example represents the system's most specific hypothesis (h_s) for the tool's structure and pose and the most general hypothesis (h_g), initially, covers everything. The system refines its hypothesis by conducting experiments. It may generalize the most specific hypothesis, then construct an instance that is consistent with its generalization, and vice versa.

To select the tool, the object that matches the most structural properties of the tool in the trainer's example is chosen. To select the pose, firstly, we collect the spatial literals in h_g then append them to the shuffled spatial predicates in h_s. Those predicates are treated as constraints and solved by a constraint solver to get numerical goal.

The learning algorithm is borrowed from Golem [Muggleton and Feng, 1990] which performs a Relative Least General Generalisation (RLGG) [Plotkin, 1971]. h_s is refined by finding the constrained Least General Generalisation (LGG) of a pair of positive examples, and h_g is refined by performing a negative-based reduction. Our tool use learning algorithm is a modification from Haber's [2015], and can be seen in our recent work [2016].

4.2 Tool Creation

The robot is provided with a simple ontology to describe the properties of simple tools. In Fig.2, this is limited to hook-like tools. Note that in the present work, this ontology is handcrafted, but it is possible to learn ontologies with ILP. The main function of our ontology is to limit the search space when a generalization is conducted. It is similar to Shapiro's "refinement graph" in MIS [Shapiro, 1983].

In tool creation, the refinement graph can be used to suggest generalizations by climbing the generalization hierarchy. For example, if the current model requires a hook to be on the right-hand side of the tool, a generalization may suggest that the hook can be on either side. In this case, the new hypothesis is tested by generating a new instance of the hypothesis that does not match previous instances seen or constructed.

Even though the refinement graph limits the search, a reasonably large graph may give rise to many possible configurations. The system could attempt to find a suitable tool by manufacturing and testing all possible tools exhaustively. To avoid this, the system prioritizes new tools that are more similar to the old one. The similarity is computed by the Levenshtein distance [Levenshtein, 1966], the number of edit operations needed to transform one representation to another.

The numerical sizes of tools are acquired by treating the structural properties as constraints and solving them. In a simulation, a tool is "manufactured" by converting its description into a URDF (Universal Robotic Description Format) file, which can be fed to Gazebo simulator. Because of a limited camera's field of view, we can only test five new tools in one batch. Tools with smaller edit distances are prioritized. If a tool use fails, the system moves on to the next one, otherwise, it checks which of the new tool's properties are different from the old one and generalize them. Our mechanism is described in Algorithm 1.

Algorithm 1 Tool creation in simulation

Require: h_s, h_g, background knowledge (BK), experiment status,learning status
1: Apply RLGG to structural properties of h_s and BK to get "generalised properties"
2: Instantiate "generalised properties" to get a collection of grounded properties of new tools
3: **for** each new tool **do**
4: Find the edit distance
5: Do constraints solving to get its numerical size
6: Create a tool in URDF
7: Select 5 novel tools with minimum edit distances.
8: Test the tool one by one.
9: **if** The testing is successful **then**
10: In old and new tools, find the literals that have same functor names, but different grounded arguments
11: Generalise the relevant properties
12: Update h_s and h_g
return Updated h_s and h_g

4.3 Explainable Tool Creation

We have an advantage in having relational representation, as its learning results are inherently interpretable by a human. Hence, to make our tool creation explainable for a human is a matter of **storing** all relevant information, including the learned hypotheses in Prolog, the primitive objects poses and the camera snapshots, **reasoning** on that knowledge base, and **presenting** them in a meaningful way to a human.

Essentially, we want the robot to have "conversation" with a human so failures can be prevented, errors can be resolved, and learning time can be reduced. There are two things that a human user can do here:

- Fully debugging the learned robot's plan in Prolog to find and correct the errors. This is following the bug-correction algorithm [Shapiro, 1983].
- Asking the robot about why it makes such a decision, and take action based on its explanations.

We will clarify these ideas on section 5.3.

5 Results and Discussions

We evaluate the CREATIVE algorithm by conducting experiments in tool use learning and tool creation. The preliminary work to demonstrate the explainability feature is also given here.

5.1 Tool use learning experiment

We perform experiments in the Gazebo simulator, which incorporates realistic physics engine [Wicaksono and Sammut,

2016]. The simulation environment includes the Baxter robot, a cube that the robot is required to retrieve from inside a tube, and five different objects that may be used as tools. We prepare two sets of five different tools based on the width of their hooks. We then randomly select the width of those tools in each trial. A downward-facing web camera is set up to capture the whole scene. A camera is also mounted in Baxter's gripper. These are required to grab the tool and pull the cube accurately.

As our primary goal is to demonstrate tool creation, we speed up our experiments by using the action models learned by Brown (2009) as background knowledge. These are then refined by experimentation. 12 learning episodes are required for the robot to learn these properties of a reliable tool [Wicaksono and Sammut, 2016]:

- The hook is attached on the same side as the cube's position inside the tube.
- The handle touches the cube on the same side as its position inside the tube.
- The hook touches the cube on its back.
- The hook is located on the same end as the cube's location inside the tube.

5.2 Tool creation experiment

Here, we change the environment to trigger the creation of novel tools, by creating a longer tube compare to the previous one. The old tools will fail as it is impossible to reach far enough into the tube, and the robot will begin to create new tools. It starts by by applying RLGG on the structural properties of the tool and the background knowledge. Only properties with grounded value that will be generalized.

The instantiation can be performed later on, so 18 potential tools are generated. We only care for the novel tools (15 tools) and ignore the old ones (3 tools). All generated tools and their edit distances are shown in Fig. 4. The black tool is old and useful, the gray tools are old but not useful, while the white ones are novel and will be tested later.

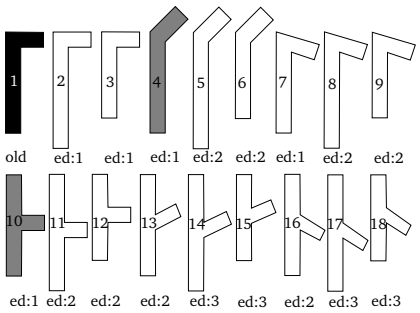

Figure 4: All generated tools and their edit distances

To select the novel tools we use two strategies: random selection and ranked selection. As the limitation of our vision sensor, only 5 tools can be tested at a time. As the result, the second strategy finds the useful novel tool, a tool with similar shape with the old one but has a longer handle (number 2), only in 3 episodes, while the first one finds it in 8 episodes.

After a useful novel tool is found, we compare its properties with the ones belonging to the old tool. To be more specific, we look for a literal with the same functor name but has different ground value. In this case, we find that `handle_length(Handle, medium)` is not valid any more and should be generalised into `handle_length(Handle, Length)`.

5.3 Experiment in explainable tool creation

Fully debugging on a learned robot's plan in Prolog is straightforward by following Shapiro's bug-correction algorithm [Shapiro, 1983]. Here, human acts as an infallible oracle, to whom the robot asks whether a Prolog predicate is correct or not. If it is incorrect, the robot will suggest the correction, and the debugging loop will be repeated until the human oracle think everything is correct.

The human-robot "conversation" on robot's decisions to create a tool is more abstract. Because of the pages limitation, we only illustrate it in a simplified and easy-to-understand form. Currently, we do not do any natural language processing, so the complete dialogue in Prolog is just using a set of simple questions and answers. The italic words are based on robot's knowledge.

ROBOT : I will print a *right_acute_hook* that has *acute_angle*, *right_side*.
HUMAN : Why you choose *right_acute_hook* ?
ROBOT : This is the successful tool after *16 experiments*. Here are *the learned hypotheses* and *the snapshots of tools and environments*.
HUMAN : Why don't you choose hook with *obtuse_angle* ?
ROBOT : I have tried *right_obtuse_hook* in *experiment 15*, but it *fails (60%)*. Do you want me to try it?
HUMAN : No, I've just checked. How about tool with *right_angle* ?
ROBOT : I have not tried *right_right_hook* yet. Do you want me to try it?
HUMAN : Yes. I think it is more likely to success compared than the one that you suggest.
ROBOT : Okay, I will try it. I will get back to you later.

Currently, this feature is only tested locally. In the future, it will be integrated into our system.

6 Conclusions and future work

We have equipped the simulated Baxter robot with an autonomous tools invention expertise. A tool ontology is developed to represent the tools and reduce the search space. We extend tool use learning, so the robot can generate novel tools and find the useful one based on the past experiences. A feature to enable robot explains its decision to human is developed, so better decision and reduced learning time can be expected.

In the future, we will build a system that combines a physical and simulated robot. Further evaluations that include another kind of tools also need to be done. Lastly, we plan to fully integrate the explainability feature into our system and improve it by providing deeper knowledge and more intuitive user interface.

Acknowledgments

Handy Wicaksono is supported by The Directorate General of Resources for Science, Technology and Higher Education (DG-RSTHE), Ministry of Research, Technology, and Higher Education of the Republic of Indonesia.

References

[Apt *et al.*, 2007] Krzysztof R Apt, Mark Wallace, et al. *Constraint logic programming using ECLiPSe*. Cambridge University Press New York, 2007.

[Beck *et al.*, 2011] Sarah R. Beck, Ian A. Apperly, Jackie Chappell, Carlie Guthrie, and Nicola Cutting. Making tools isn't child's play. *Cognition*, 119(2):301 – 306, 2011.

[Brodbeck *et al.*, 2015] Luzius Brodbeck, Simon Hauser, and Fumiya Iida. Morphological evolution of physical robots through model-free phenotype development. *PLOS ONE*, 10(6):1–17, 06 2015.

[Brown and Sammut, 2012] Solly Brown and Claude Sammut. A relational approach to tool-use learning in robots. In *Inductive Logic Programming*, pages 1–15. Springer, 2012.

[DARPA, 2016] DARPA. *Broad Agency Announcement: Explainable Artificial Intelligence (XAI)*, 2016.

[Fikes and Nilsson, 1971] Richard Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artif. Intell.*, 2(3/4):189–208, 1971.

[Haber, 2015] Adam Haber. *A system architecture for learning robots*. PhD thesis, School of Computer Science and Engineering, UNSW Australia, 2015.

[Levenshtein, 1966] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.

[Mar *et al.*, 2015] Tanis Mar, Vadim Tikhanoff, Giorgio Metta, and Lorenzo Natale. Self-supervised learning of grasp dependent tool affordances on the icub humanoid robot. In *Proceedings of ICRA*, May 2015.

[Mitchell, 1977] Tom M Mitchell. Version spaces: A candidate elimination approach to rule learning. In *Proceedings of the 5th international joint conference on Artificial intelligence-Volume 1*, pages 305–310. Morgan Kaufmann Publishers Inc., 1977.

[Muggleton and Feng, 1990] Stephen Muggleton and Cao Feng. Efficient induction of logic programs. In *New Generation Computing*. Academic Press, 1990.

[Muggleton, 1991] Stephen Muggleton. Inductive logic programming. *New Generation Computing*, 8(4):295–318, 1991.

[Plotkin, 1971] G. D. Plotkin. A further note on inductive generalization. In B. Meltzer and D. Michie, editors, *Machine Intelligence 6*. Elsevier, New York, 1971.

[Rosenthal *et al.*, 2016] Stephanie Rosenthal, Sai P Selvaraj, and Manuela Veloso. Verbalization: Narration of autonomous robot experience. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 862–868. AAAI Press, 2016.

[Sammut, 1981] C. A. Sammut. Concept learning by experiment. In *Seventh International Joint Conference on Artificial Intelligence*, pages 104–105, Vancouver, 1981.

[Shapiro, 1983] Ehud Y Shapiro. *Algorithmic program debugging*. MIT press, 1983.

[Sheh, 2017] Raymond Sheh. "why did you do that?" explainable intelligent robots. In *AAAI Workshop on Human-Aware Artificial Intelligence*, 2017.

[Stoytchev, 2005] A. Stoytchev. Behavior-grounded representation of tool affordances. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 3060–3065, April 2005.

[Tikhanoff *et al.*, 2013] V. Tikhanoff, U. Pattacini, L. Natale, and G. Metta. Exploring affordances and tool use on the icub. In *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 130–137, Oct 2013.

[Wang *et al.*, 2014] Liyu Wang, Luzius Brodbeck, and Fumiya Iida. Mechanics and energetics in tool manufacture and use: a synthetic approach. *Journal of The Royal Society Interface*, 11(100), 2014.

[Wicaksono and Sammut, 2016] Handy Wicaksono and Claude Sammut. Relational tool use learning by a robot in a real and simulated world. In *Proceedings of ACRA*, December 2016.