Goal-Directed Procedure Learning

Patrick Hammer[1(✉)] and Tony Lofthouse[2]

[1] Temple University, Philadelphia, USA
patham9@gmail.com
[2] Evolving Solutions, Andover, UK
tony.lofthouse@gmilab.com

Abstract. A novel method of Goal-directed Procedure Learning is presented that overcomes some of the drawbacks of the traditional approaches to planning and reinforcement learning. The necessary principles for acquiring goal-dependent behaviors, and the motivations behind this approach are explained. A concrete implementation exists in a Non-Axiomatic Reasoning System, OpenNARS, although we believe the findings may be generally applicable to other AGI systems.

Keywords: Goal-directed · Procedure learning · Preconditions
Motor-functions · Non-Axiomatic Reasoning System

1 Introduction

Acquiring procedural knowledge is generally concerned with representing the preconditions and consequences of actions. In traditional planning approaches [11], this knowledge is usually provided in advance, and the task is then to search for the most concise and complete plan that leads to the achievement of a certain goal, a desired end state. This approach can be modified to search for the plan that leads to the end state with the highest probability. The drawback to this approach is that the need to react to change of circumstance during the planning and execution process, as well as forming the preconditions and transition probabilities from experience, is not captured.

In Reinforcement Learning (RL) this problem is reduced to learning to act the right way in the currently observed situation, where "act the right way" is usually taken as the selection of the action with the maximum expected utility value [12]. Here, no explicit plan is generated, and no subgoal derivations happen. Instead the decision-making is only considering the currently observed state, whilst assuming it is a complete description of the current situation [12]. While being sufficient in applications where the system's behavior serves a single purpose, this treatment becomes insufficient when novel goals spontaneously appear or existing ones disappear [16]. That's clearly the case in many robotics applications, [7], and also, as many argue, in the human mind [4,9]. To improve the ability to adapt to changing circumstances, a change of goals should not require re-learning the related situation-action mappings. Instead an AGI system should develop beliefs of the consequences of its actions in such a way, that when goals change, the actions that lead to the fulfillment of novel goals can be derived. This can be seen as an understanding of the environment in terms of how different outcomes can be achieved by the system, independently from what is currently "rewarded" or desired.

In this paper we present a method that combines the benefits of both traditional planning and RL, while eliminating some of the drawbacks of both approaches. As our approach is based on learning the preconditions and consequences of operations, there is no need to re-learn situation-action mappings when goals change, a major advantage. In our approach, each piece of procedural knowledge can be independently evaluated and suggests a certain operation in a certain context to realize a certain subgoal: solving "global" problems, using "local" decisions. So, different to traditional planning approaches, no complete plan is explicitly searched for, instead the individually evaluated pieces lead to a certain behavior, influenced by current goals and observations.

2 The Goal-Directed Procedure Learning Problem

We regard procedure learning as the process that forms procedural hypotheses, based on temporal patterns of events and operations that appear in the systems experience. A temporal pattern can be represented as $(A_1, ..., A_n)$, which is a sequence of n consecutive events or operations, each occurring at a certain discrete time-step. Here, events do not encode an entire state, just certain parts of it, such as temperature information coming from a sensory device, encoded by a composition of terms/IDs, called "Compound Term" (see [14]). Now, temporal patterns can become building blocks of hypotheses, which should capture useful regularities in the experience of a system. In general, a hypothesis can be defined as $A \Rightarrow B$, with a special case, where a procedural hypothesis can be defined as $(A, B) \Rightarrow C$ where A is an antecedent, B an operation and C a consequent. The antecedent can be considered as a precondition that, when followed by an operation, is believed to lead to the consequent. Additionally, with the inclusion of temporal constraints, hypotheses can be considered as predictive, such as $(A, B)\ /\!\!\Rightarrow C$, whereby they imply the occurrence time of the consequent to be in the future. Here, the precondition and consequent can be arbitrarily complex pattern compositions, while the behavior is usually an atomic, in the sense that it can be directly invoked. Additionally, the consequent often represents a goal or subgoal to realize. Furthermore each such hypothesis has a degree of certainty corresponding to the likelihood that the prediction will be confirmed when its precondition is fulfilled.

Given a goal $G!$, which a system wants to make as true as possible, how does it satisfy the goal? [15].

There are two approaches: via hypotheses formation through forward chaining, where helpful hypotheses are formed directly from pieces of knowledge, with observed patterns as special cases, and backward chaining, where a subgoal is derived from a goal and an existing piece of knowledge [15].

In the case where a hypothesis exists, for how to achieve $G!$ in the current context, usually a single backward chaining step to derive the operation as a subgoal, which can be directly executed, will be sufficient. Therefore, "executing a procedural hypothesis", means to perform a single backward inference leading to an operation, that can be executed.

Generally, problem-solving involves an inter-play between "chain" and "execute". That is, because when the system has no appropriate procedural hypothesis, that both predicts G and also has its preconditions currently met, then usually both chaining strategies will be necessary, to search and probe for a solution, a schema that can later be re-used [3]. The "chain" case requires creative but evidence-driven exploration of possibilities to deal with novel situations. This corresponds to finding a solution although no algorithm is known, as discussed in [15]. The "execute" case on the other hand requires the control mechanism to remember solutions to goals. This allows for faster response times to similar problems in the future and to make effective use of what has already been learned.

We believe that most of the things we do, such as driving a car, that increasingly begin to become more automatic, are due to a transition from the "chain" to the "execute" case [2]. A transition from the "novel" to the "usual" is something a systems control mechanism can and should effectively support. Furthermore, a control mechanism, will need to choose between multiple hypotheses which can satisfy a goal, to a different degree, for the current context. Therefore the formation, selection and testing of hypotheses are the main topics of this paper and each of these aspects will be described in detail below.

3 OpenNARS Considerations

A goal of this paper is to present our findings in as general a way as possible, so as to allow for the widest applicability as possible. Notwithstanding this goal, our methods, as presented in this paper, have a concrete implementation in the OpenNARS system. An explanation of some of the data structures and conceptual ideas will be of value in understanding the approaches outlined below. In particular, four aspects will be described at a relatively high level of abstraction, namely: evidence, budget, concepts, and bags. Detailed explanations of each of these can be found here [5,13,14].

Evidence. In NARS evidence is used to provide the truth of a belief, namely its certainty. It is defined as a (w_+, w_-) pair, where w_+ represents positive evidence, and w_- represents negative evidence, or alternatively as confidence c and frequency f tuple, where $f = \frac{w_+}{w_+ + w_-}$ and confidence is $c = \frac{w_+ + w_-}{k + w_+ + w_-}$, where k is a global personality parameter that indicates a global evidential horizon [6]. Evidence supports these principles:

- An item of evidence can only be used once for each statement.
- A record of evidence (a set), used in each derivation must be maintained, although this is only a partial record due to resource constraints, which is not an issue in practice.

- There can be positive and negative evidence for the same statement.
- Evidence is not only the key factor to determine truth, but also the key to judge the independence of the premises in a step of inference. Inference is only allowed when the evidence records of the two premises do not overlap. This not only avoids cyclic inference, but also keeps revision from further increasing the confidence of a belief on re-derivation.

So we can now define the degree of certainty of a hypothesis, when it's precondition is fulfilled, as a *truth* value as defined above. Therefore, the positive evidence for a predictive hypothesis is simply a measurement of how frequently the occurrence of the antecedent was followed by its consequent event, and the negative evidence how frequently the consequent did not happen whilst the antecedent did. Truth expectation, which we will sometimes refer to, 'merges' frequency and confidence into a single value, that can be used for comparison purposes. For detailed formulas, see [5,14].

Budget. The amount of system resources, namely CPU and memory, that is allocated to a specific task, an item of work, is directly related to budget, which is a tuple (p, d, q) where p is priority, d is durability and q is quality, and all parameters are between 0 and 1. Priority determines short term importance whilst durability determines long term importance. Quality is effectively implemented as a priority barrier (usually between 0 and 0.1) under which the priority value can not fall. This ensures that items of high long-term importance will survive in the bag even though they have a low priority. More details can be seen in [5]. Besides quality, there are many factors that affect the priority of an item [13], such as, but not limited to:

- How recent the event is (is it still relevant?)
- How often does the event occur (how stable is it?)
- Did the event happen unexpectedly? (how surprising was its occurrence?)
- Is it related to a goal or question?

Bags. One of the constraints of an AGI system is that it needs to work with finite resources. When working within a fixed sized memory, once a memory limit is reached, a decision has to be made as to which item to remove to make space for a new one. Here, a data structure called "Bag" is used by OpenNARS. This data structure stores items ordered by their priority value, allowing for sampling items based on the priority distribution within. Once a bag is full, in order to make room for a new item, the lowest-priority item is removed. Bag is constructed in such a way as to support efficient sampling, adding and removal of items without any search operations, so all operations on bag are O(1). Overall this control strategy is very similar to the Parallel Terraced Scan in [10], as it also allows the exploration of many possible options in parallel, with more computation devoted to options which are identified as being more promising. OpenNARS controls the resource allocation of all reasoning using the bag data structure, and is not restricted to reacting to events. The aforementioned chaining case is also controlled by Bag sampling. See [13] for more detail.

Concepts. Conceptually, the belief network of the OpenNARS system can be thought of as a directed graph where vertices are *concepts* and edges are *links* (links are outside the scope of this paper (see [5]). For the purpose of this discussion concepts can be considered as storage units, that contain a bag of beliefs and goals. Concepts themselves are stored in a large system-wide *ConceptBag* and selected as explained above. Therefore, concepts, beliefs and goals are selected probabilistically, based on their priority and forgotten when their priority is the lowest in the respective bag.

4 Goal-Directed Procedure Learning: Method

Temporal Reasoning. An adaptive agent existing in a real-time environment needs to be aware of, and capable of reasoning about, time. We refer to this mechanism as Temporal Reasoning. An event is something that the system experiences in *time* and this time is captured as an *OccurrenceTime*. In OpenNARS occurrence time is measured in system cycles, but can be any representation that supports a regularly incremented value, such as a real-time clock. When reasoning with time the notion of 'interval' becomes important. For example, two events occurring time t apart can be represented as (E_1, I_t, E_2), where I_t is an interval of time duration t. This allows arbitrarily complex temporal patterns to form. These patterns form the necessary preconditions for hypothesis creation. Intervals also apply to implications, $(E_1, I_t) \not\Rightarrow E_2$, where the temporal aspect becomes part of the precondition. Intervals are always measured, though we will omit them in the discussion whenever they don't add any value. A key challenge is how to allow intervals of different duration to be revised. Imagine two hypotheses $(E_1, I_{t1}) \not\Rightarrow E_2$ and $(E_1, I_{t2}) \not\Rightarrow E_2$. Both of them predict the same outcome based on the same precondition, but they expect different interval durations, I_{t1} and I_{t2}. To allow these different intervals to be revised, a confidence decay (Projection) [5] increasing with the time difference is applied after revision. Here it makes a difference which premise is projected to the other. The projected premise should be the one whose timing appears less often, so as to keep the more usual timing in the conclusion. This enables the learning of the more, commonly experienced, interval durations over time.

Hypothesis Creation. This is the core of the method, and whilst selection and testing are important, creation is the key to building relevant and useful hypotheses. The crucial insight was to separate the incoming experience stream into events and operations (in OpenNARS operations are restricted to the events that the system can initiate itself. Operations also generate input events as feedback). With this separation, the task of forming meaningful preconditions was a simpler problem to solve: operations simply become the context under which certain events cause others to occur. Given, a collection (a Bag in OpenNARS) of recent *Events* and, a collection of recent *Operations*, a hypothesis is formed in the following way:

1. Probabilistically select an Op from *Operations*, where the probability of selection should be roughly proportional to how long ago the event happened or the operation was invoked. After invocation of an operation:
   (a) Create a concept for the Op (if it does not already exist) [5].
   (b) Copy the *Events*, that occurred prior to the operation, based on occurrence time, to the Op concept.
   (c) The Op concept can now form preconditions, based on probabilistic selection of these events, and construct premises of the form (E, Op), where E is a precondition of the Op.
2. Now when a new event E^* enters the system and an operation Op is sampled from *Operations*, two steps occur:
   (a) Sample a second premise E_{past} from *Events* to form temporal sequences (E_{past}, E^*) that are inserted into *Events*, and also the predictive hypothesis $(E_{past} \; /\!\!\Rightarrow \; E^*)$ which only exists in concept memory. The latter is required as not everything can be understood in terms of own operations (as also argued by [1]), such as the transition from day to night.
   (b) Based on Op, retrieve, via probabilistic sampling of the Op concept, one of the (E, Op) preconditions, in order to form a procedural hypothesis $((E, Op) \; /\!\!\Rightarrow \; E^*)$. Here, clearly the consequent is put into the context of the operation. Without taking operations to execute into account, there is simply no way to predict, for example, where the cup of tea in your hand will move next, as argued by [1].

Over time the total evidence of re-occurring hypotheses will increase due to the Revision rule [14] being applied within the concept of the hypothesis.

Revised procedural hypotheses $((A, Op) \; /\!\!\Rightarrow \; B)$ are also stored in the "foreign" concept B, this allows B to memorize ways of how it can be realized, and to learn its preconditions. Thereby, the most successful hypotheses, these with the highest truth expectation, will become the most likely to be selected.

Hypothesis Selection. Assuming an incoming or derived goal $G!$, the task of hypothesis selection is to choose the most relevant hypothesis that can satisfy $G!$ with some previously experienced event E as precondition. Also assuming that such a hypothesis already exists, the Detachment rule [14] can be applied twice to a matching hypothesis, which is of the form $(E, Op_i) \Rightarrow G$. The first detachment leads to $(E, Op_i)!$ and the second one to $Op_i!$. The $Op_i!$ with the highest truth expectation, or certainty, will most likely lead to the greatest satisfaction of $G!$, and therefore will be derived. $G!$ can then be revised in concept G and trigger an execution if the truth expectation of the revised goal (projected to the current time) will be above a decision threshold [5]. Note that subgoal derivations also happen in the backward chaining process we described. This is especially important when no hypothesis that can directly realize $G!$ exists, or new solutions should be probed for.

Hypothesis Pruning. Given the uncertain nature of input experience, it is not possible, in advance, to identify what will be relevant and useful. This, unavoidable, lack of foresight can lead to the formation of hypotheses that have little value to a system. Additionally, given the limited computational resources, only a certain number of hypotheses can be retained at any one time. This makes it even more important to keep track of the success rate of hypotheses, as to keep the most competent ones while removing the others.

The approach taken to allow for this is hypothesis pruning, to measure the success of hypotheses so that these that do not predict correctly can be removed by lowering their quality. While finding positive evidence is achieved through temporal induction rules as mentioned before, finding negative evidence is the job of Anticipation: given a predictive statement of the form: *antecedent* $/\!\!\Rightarrow$ *consequent*, we define *Anticipation* as the expectation that the antecedent will lead to the consequent being observed as predicted. With Anticipation a system is able to find negative evidence for previously learned predictive beliefs which generate wrong predictions [5,8].

If the event happens, in the sense that a new input event with the same term as the anticipated event is observed, the anticipation was successful (Confirmation), in which case nothing special needs to be done, since the statement will be confirmed via the normal process of temporal induction.

If the predicted event does not happen then the system needs to recognize this. This is achieved by introducing a negative input event, $not(a)$. Note that in this case, such a negative input event has high priority and should significantly influence the attention of the system, assuming such a mechanism is present.

Here, one challenge is how to determine the timeout duration after which we decide the prediction failed. A simple treatment turned out to be effective: Given that an event was predicted to occur in n steps, the failure can be recognized after a certain multiple of n steps. A more refined approach would be to keep track of the variance in timings in the concept of the predictive hypothesis, and then to decide that failure point taking the variance into account. But also in this case, a decision as to where to set the failure point, has to be made.

5 Results

Test Chamber is one of the environments that we developed to allow an AGI system to be tasked with a variety of different goals and experiences in novel surroundings. Within Test Chamber an AGI is expected to demonstrate goal-oriented observation-based procedure learning in a domain of doors, switches and lights. A birds eye view perspective is controlled by a user, opening doors, picking keys, and so on. The system observes the users actions and can call different operations directly, such as: going to an object, activating a switch, deactivating it, picking an object from the floor and so on. Activating operations in different contexts allows for different outcomes in the test environment.

A possible event stream generated from observing a user:

1. Event E_1: Reached start place
2. Operation Op_1: Go to switch1
3. Event E_2: Reached switch1
4. Operation Op_2: Activate switch1
5. Event E_3: Switch1 activated

Although these examples were performed in OpenNARS, for the purpose of this paper it is sufficient to know that these inputs can be directly represented as events and operations, and potentially in other systems.

OpenNARS easily creates hypothesis $((E_1, Op_1) \mathbin{/\!\!\Rightarrow} E_3)$ by making use of the explained mechanisms. The same is true for $((E_2, Op_2) \mathbin{/\!\!\Rightarrow} E_3)$. While there are other generated results too, these two are of special relevance: assuming a goal E_3 exists or enters the system, what hypothesis, or effectively behavior, should the system choose to reach its goal? It depends on the context. In this particular example, both hypotheses effectively help each other and allow for "local" decisions to realize E_3 dependent on whether E_1 or E_2 was observed: when one of the hypotheses doesn't exist, the other one would fail, as an important part of the necessary behavior would be missing, only together can they succeed. A surprising property of this implicit representation is that no explicit "global plan" exists or needs to be searched for, yet is totally sufficient for carrying out the task successfully. However, when an important piece of knowledge is missing, a fast reaction is often not possible. In this case, a system needs to improvise, either by searching for a solution in its memory, or by probing the environment, both of the strategies should be applied together rather than in a distinct way.

These representative cases work well in OpenNARS, and allow it to, more easily, acquire complex behaviors whilst switching its behavior when goals and context change.

We also applied OpenNARS to Reinforcement Learning problems, one of them being Pong: assuming a goal $G!$, an input event $G.$ can directly act as "reward signal", which in Pong basically is an event encoding "the ball collided with the bat". Additionally, the system is given events about the horizontal ball position relative to the bat. This allows it to invoke different operations dependent on whether the ball is left or right of the bat. For this purpose, it can invoke two operations: to move the paddle to the left or to the right. This turned out to be sufficient to let the system learn to Play Pong in short time and with high reliability: the experiment was repeated 50 times, and the system learned the right behavior in all of the cases, with 98 s mean and a variance of 51 s until the right policy was learned. Important to note here is that the ball starts at a random position, with a random movement direction, and needs at least 5 s to reach another side of the quadratic board. This explains the high variance, as some representative cases need to occur first, before the relevant hypotheses will be supported by real evidence. Another factor here: when a wrong behavior (such as moving to the left side when the ball is on the right side) is learned first, due to unlucky cases, it will take longer for the right hypothesis to take over, as negative evidence generated by failed anticipations needs to be found before the right behavior will be tried.

6 Conclusion

The Reasoning-Learning Mechanism employed by NARS has been shown to be capable of goal-directed Procedure Learning: the separation of operations from other events has enabled the system to form more successful and useful hypotheses with little resource effort. These are subsequently used to enable fast hypothesis selection through the precondition memorization mechanism. This mechanism allows the system to make effective use of procedural knowledge and have fast response times when the relevant knowledge already exists. Collectively our methods allow the system to self-program and automatize itself to become gradually more competent over time. We have shown that the system can learn goal-oriented procedures involving multiple operations, without building explicit plans. Furthermore, we have demonstrated, that the system can perform well in Reinforcement-Learning style tasks as a special case, and that the "reward signal" can naturally be represented. While this paper was mainly about introducing the involved mechanisms and their properties, future work will include detailed comparisons with alternative procedure learning techniques. This will include theoretical comparisons as well as detailed results on learning performance.

In conclusion, we believe the techniques presented in this paper, specifically, operational separation, precondition memorization and Anticipation are generally applicable to a broad class of AGI systems. As has been highlighted above, the Procedure Learning capability of OpenNARS was significantly improved as a result of these enhancements.

References

1. Ahmad S., Hawkins J.: Untangling Sequqnces: Behaviour vs External Causes, bioRxiv (2017)
2. DeKeyser, R.: Skill acquisition theory. In: Theories in Second Language Acquisition: An Introduction, pp. 97–113 (2007)
3. Drescher, G.: Made-up minds: a constructivist approach to artificial intelligence. Ph.D. thesis, MIT, Computer Science, September 1989
4. Eagleman, D.M., Sejnowski, T.J.: Motion integration and postdiction in visual awareness. Science **287**, 2036–2038 (2000)
5. Hammer, P., Lofthouse, T., Wang, P.: The OpenNARS implementation of the non-axiomatic reasoning system. In: Steunebrink, B., Wang, P., Goertzel, B. (eds.) AGI -2016. LNCS (LNAI), vol. 9782, pp. 160–170. Springer, Cham (2016)
6. Pöppel, E., Bao, Y.: Temporal windows as a bridge from objective to subjective time. In: Arstila, V., Lloyd, D. (eds.) Subjective Time: The Philosophy, Psychology, and Neuroscience of Temporality. The MIT Press, Cambridge (2014)

7. Latombe, J.C.: Robot Motion Planning, vol. 124. Springer, New York (1991). https://doi.org/10.1007/978-1-4615-4022-9
8. Nivel, E., et al.: Autocatalytic Endogenous Reflective Architecture (2013)
9. Pitti, A., Braud, R., Mahé, S., Quoy, M., Gaussier, P.: Neural model for learning-to-learn of novel tasks in the motor domain. Front. Psychol. **4**, 771 (2013)
10. Rehling, J., Hofstadter, D.: The parallel terraced scan: an optimization for an agent-oriented architecture. In: Proceedings of the IEEE International Conference on Intelligent Processing Systems 1997 (1997)
11. Russel, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall Series on Artificial Intelligence. Prentice Hall, Englewood Cliffs (2009)
12. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (2017)
13. Wang, P.: Rigid Flexibility: The Logic of Intelligence. Springer, Dordrecht (2006). https://doi.org/10.1007/1-4020-5045-3
14. Wang, P.: Non-Axiomatic Logic: A Model of Intelligent Reasoning. World Scientific, Singapore (2013)
15. Wang, P.: Solving a problem with or without a program. J. Artif. Gen. Intell. **3**(3), 43–73 (2013)
16. Wang, P., Hammer, P.: Assumptions of decision-making models in AGI. In: Bieger, J., Goertzel, B., Potapov, A. (eds.) AGI 2015. LNCS (LNAI), vol. 9205, pp. 197–207. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21365-1_21