

A Preliminary Collection of Reflective Critics for Layered Agent Architectures

Push Singh

MIT Media Lab, 20 Ames St. Cambridge, MA 02138, USA
push@media.mit.edu

Abstract. How can we build more robust reasoning systems? One approach is to build systems as layered agent architectures where each layer observes the activities occurring in the layers beneath, and uses knowledge about how to interpret these observations to control and debug those activities. Architectures with such ‘towers of reflection’ are used because it is often difficult to assure perfect operation in any one layer, and therefore an additional layer that reflects on it can be added to help cope with bugs within lower layers. In this paper we consider three layer architectures, ones with reactive, deliberative, and reflective layers. In the deliberative layer, any particular method of inference, strategy for producing a plan, or other method of deliberation is likely to break now and then. We propose that *reflective critics* are a primary type of agent in the reflective layer that is concerned with noticing problems in the recent activities within the deliberative layer. In this paper we explore what types of reflective critics there are and how they can be represented declaratively. We illustrate the use of these reflective critics with a set of scenarios drawn from problems encountered during typical computer use. While this work is still preliminary, we believe it represents a step towards more self-reflective and self-repairing agent systems.

1 Introduction

How can we build more robust reasoning systems? One approach is to build systems as layered agent architectures where each layer observes the activities occurring in the layers beneath, and uses knowledge about how to interpret these observations to control and debug those activities [1] [2]. Architectures with such ‘towers of reflection’ are used because it is often difficult to assure perfect operation in any one layer, and therefore an additional layer that reflects on it can be added to help cope with bugs within lower layers.

The most common such architectures consist of three layers: the reactive, deliberative, and reflective layers. The reactive layer acts to sense and effect the external world, the deliberative layer models and produce inferences about the capabilities of the reactive layer and the external world, and the reflective layer in turns models the inference processes that happen in the deliberative layers and how they influence the reactive level and in turn the external world.

The deliberative layer is populated with processes that produce inferences contributing to various cognitive goals such as planning, recognition, prediction, explanation, and other modes of deliberation that are based on using models of the reactive layer and the external world. However, in a complex enough system, it is likely that such deliberative processes will contain bugs—any particular item of knowledge, method of inference, strategy for producing a plan, or other ingredient of deliberative thinking could be flawed. Perhaps an item of knowledge, while often true, fails to apply in the current situation; or perhaps because the world is not fully observable, an incorrect inference is made about the current state of affairs; or perhaps a plan of action fails to take into account certain negative consequences. There are many ways to ‘think badly’!

In this paper we describe some preliminary steps we have taken towards declaratively representing such ‘thinking bugs’, in order to incorporate into the reflective layer powerful ways to reflect upon and debug the processes in the deliberative layer. We introduce the term *reflective critic* for the special class of agents concerned with noticing problems with recent activity within the deliberative layer. Here are some examples of the kinds of observations that such reflective critics might produce about recent deliberations:

- *I was working on a solution using a difficult method, and then realized that a simpler method had been available to me all along.*
- *I took an action, but afterwards realized that I should not have because it had a latent negative side effect, and the effects of that action turned out to be undoable.*
- *I had assumed that certain objects would be obstacles, but they turned out not to be and in fact they were helpful.*

In this paper we explore what types of reflective critics there are, how they can be represented, and how they might be incorporated into a layered agent architecture.

This paper is structured as follows. We first review some of the history behind the notion of a ‘critic’, and clarify the role that reflective critics in particular play within an agent architecture. We then describe a language of mental events for describing activities within the deliberative layer of an architecture, use that language to provide some examples of how reflective critics could be represented, and then illustrate the use of these reflective critics with a set of scenarios drawn from problems encountered during typical computer use. We conclude by reflecting on the notion of a reflective critic and discuss where we plan to go next with this idea.

2 Critics in existing systems

The notion of a ‘critic’ that embodied knowledge about bugs in programs was first explored by Gerald Sussman in his Ph.D. thesis [3]. Since then, this has become a relatively common technique in automated planning systems. However, critics have seen little use in other areas of reasoning. Sussman’s original suggestion was that critics were so important a type of knowledge that there should be a research program to develop a large catalogue of such critics. Such a catalog has never been made,

presumably because computer scientists have focused largely on analyzing ‘correct’ programs rather than the partial solutions and ‘nearly correct’ programs that our programs really are during most of the course of their development. However, there has been more specific work on developing the kinds of knowledge that the reflective layers of an architecture needs in order to reflect effectively enough to improve current and future deliberations.

One example is the REFLECT project [4], which explored the kinds of reflective competences that a ‘competent’ system should exhibit. For example, it should be able to reflect on the feasibility of a solution, whether it has been given contradictory requirements, whether the requirements are redundant, and so forth. The REFLECT project explored the use of ten kinds of meta-competences that could analyze and repair solutions to problems.

In the case-based reasoning community, several systems have incorporated some reflection within the case-based reasoning process. Fox and Leake [5] explore how to incorporate introspection into a case-based planning system to give it the capacity to improve the way it indexes its cases. Their ROBBIE system uses introspective reasoning to monitor the retrieval process of a case-based planner in order to detect retrieval of inappropriate cases, and when retrieval problems are detected, it explains the source of the problems and uses those explanations to determine new indices to use for future case retrieval. Also, the AUTOGNOSTIC problem solving system of Stroulia and Goel [6] makes use of a detailed model of its own problem solving methods. It uses of a variety of critics that, for example, recognize that its knowledge is incomplete or incompletely organized, in order to make repairs to itself.

Within the cognitive architectures community, perhaps the most well-known variant of the reflective critic idea is the concept of an "impasse" in the Soar cognitive architecture [7]. Soar is a rule-based system that has four basic types of reflective critics: (a) no rule matches, (b) several rules match and there is no criteria for choosing between them, (c) there is conflicting criteria for choosing between multiple rules, and (d) all matched rules are rejected by our criteria.

Each of these systems explored a few of the kinds of reflective critics one might use during reasoning. In our view, these represent a small subset of the great range of potential reflective critics, and we are attempting to assemble a large and reasonably fine-grained ontology of the kinds of bugs that show up in reasoning systems. There are a tremendous number of ways things can go wrong in the deliberative layer: an idea could require some resource that is impossible to obtain; it could have a debilitating side effect; it could have a low probability of working; and so forth.

3 The role of reflective critics within a layered architecture

We envision an architecture incorporating reflective critics as organized roughly along the lines shown in Figure 1 below. In this diagram, the agents in the reactive level represent physical actions like *send document X to printer Y*; the agents in the deliberative level represent cognitive actions like *retrieve a script where we once solved a similar problem*; and the agents in the reflective level are either *impasse critics* that recognize that an action in the reactive level is failing or has failed, *reflec-*

tive critics that recognize that a cognitive action in the deliberative level has failed, or *reflective debuggers* that respond to critics by making modifications or repairs to agents within either the deliberative or reactive layers. In this paper we focus on reflective critics.

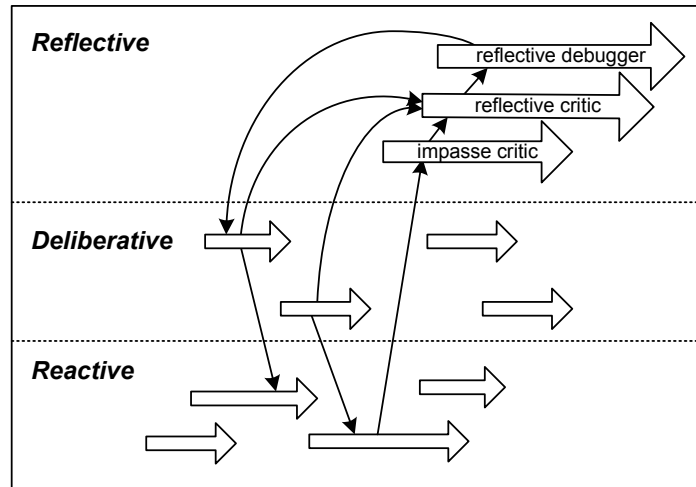


Figure 1. The structure of a three layer agent architecture. The thick arrows represent agents operating within each architectural layer, and the thin arrows represent communication between those agents.

4 Representing reflective critics

In this section we give some examples of how reflective critics can be represented. We first describe a language of mental events that we use to represent activity in the deliberative level; next, we use these to assemble a variety of simple reflective critics; then, we present a longer list of reflective critics described more informally in English.

4.1 Representing mental events

In order to encode our reflective critics, we first need to establish what the reflective layer can observe within the deliberative layer. We need an ontology of *mental events* [8] rich enough to describe the important activities in the deliberative level. The ontology of mental event predicates that we use to represent reflective critics in the following examples is shown in Table 1. In this section we take the view that reflective critics are recognizers of particularly unproductive classes of mental events expressed in terms of these predicates.

Table 1. An Ontology of Mental Event Predicates

Before(t1, t2)—time t1 < time 2
Believe(x, t)—x is believed at time t
Better(s1, s2)—situation s1 is closer to the goal than s2
Consecutive(t1, t2, t3)—time t1 < time t2 and time t2 < time t3
Do(m, t)—method m is applied at time t
DoDuring(m, t1, t2)—method m is applied between time t1 and time t2
Easier(m1, m2)—method m1 is simpler to apply than m2
EffectOf(m, eff)—method m has effects eff
Fails(a, t)—action a fails at time t
FailsDuring(g, t1, t2)—attempts to achieve goal g fail between time t1 and t2
Goal(g)—g is a goal situation
Includes(a, b)—partial state a is true within partial state b
KindOf(a, b)—thing a is a member of class b
Obstacle(o, a)—object o is an obstacle to an action a
Realize(x, t)—infers x at time t (unknown before)
Situation(s, t)—the situation s held at time t
Subgoal(g1, g2)—g1 is a subgoal of g2
Suitable(m)—method m is applicable and should achieve the current goal
Successful(m)—method m succeeded
TryToAchieve(g, t1, t2)—tried to achieve goal g between times t1 and t2
Undesirable(p)—property p is undesirable

4.2 Examples of reflective critics

We first describe each reflective critics in English, and then provide one example of how they might be represented more formally so that recognizers for such criticisms can be programmed. It should be noted that the following examples are not meant as the best or final way to represent reflective critics—presumably each critic could be represented in a variety of different ways that makes use of different ways of representing time, causality, events, goals, and so forth.

No-Past-Success. This critic notices that a method that is currently being applied has never in the past succeeded.

```
No-Experience ←  
  Do(method1, Now),  
  KindOf(method1, methodType),  
  Do(method2, t),  
  KindOf(method2, methodType),  
  Before(t, Now),  
  Not(Successful(method2)).
```

Mistaken-Opportunity. This critic notices that the system was working on a solution, but the circumstances allowed it to quickly try something different. However, that failed, and unfortunately it also undid what progress it had made using the original method.

```
Mistaken-Opportunity ←  
  DoDuring(method1, t1, t2),  
  Realize(Suitable(method2), t),  
  Consecutive(t1, t, t2)  
  Do(method2, t2),  
  Situation(s1, t),  
  Fails(method2, t3),  
  Situation(s2, t3),  
  Better(s1, s2).
```

Another-Method-Available. This critic notices that the system was working on a solution using a difficult method, then realized that a simpler method had been available to it during that period.

```
Another-Method-Available ←  
  Suitable(method1),  
  DoDuring(method1, t1, t2),  
  Suitable(method2, t),  
  Easier(method2, method1),  
  Consecutive(t1, t, t2).
```

Undoable-Negative-Side-Effect. This critic notices that the system took an action it should not have, because that action had a latent negative side effect that turned out to be undoable.

```
Undoable-Side-Effect ←  
  Do(action, t1),  
  EffectsOf(action, eff),  
  Realize(Includes(eff, sideEff), t2),  
  Undesirable(sideEff),  
  TryToAchieve(Not(sideEff), t2, t3),  
  FailsDuring(Not(sideEff), t2, t3),  
  Consecutive(t1, t2, t3)
```

False-Subgoal. This critic notices that the system was distracted by a subgoal that did not help it achieve any of its more important goals.

```
False-Subgoal ←  
  Goal(goal, t1),  
  Subgoal(subgoal, goal, t1),  
  Situation(s1, t1),  
  Situation(s2, t2),  
  Before(t1, t2),  
  Better(s2, s1).
```

5 Usage scenarios

We present below a collection of scenarios that illustrate possible contexts in which reflective critics may be used. We consider the kinds of computer-related problems that typically occur in the office context (such as a printer running out of paper).

Inaccessible-Resource. The system has assumed that a resource can be used to perform some function, but it turns out that the resource cannot be fully accessed.

Example: The system prints to the nearest printer—the office next door—but it turns out that this printer is in a locked office.

Mistaken-Obstacle. The system expected certain objects or agents to be obstacles, but they turned out not to be and in fact they were helpful.

Example: On a busy printing day someone else managed to send their document to the central printer before you could. However, it turns out they only used it briefly and then reloaded the paper tray.

Wrong-Order. The system fails at several attempts to solve a problem. It realizes that if it had tried the last attempt first, it might have worked.

Example: The system attempts to print to a printer but there is no available printer driver. It then tries to download the driver from the web site but the web site is down. It then tries to ask someone else who has a similar printer but they have already left for the day.

Undoable-Action. The system performed an action that could not be undone, even though it had expected it could be.

Example: The system backed up the file on a portable hard drive and then deleted it off the current drive to save space. However, the portable drive was then lent to someone for a business trip, leaving the file inaccessible.

Undoable-Replacement. The system needs to replace a component. However, after it removes the original component it realizes that the new component is no longer available. The old component cannot be replaced and the system is left non-functional.

Example: The system attempted to install several libraries needed by a Linux application. However, this required first uninstalling the previous libraries. The new libraries turned out not to be downloadable from any existing download site. The old libraries cannot be reinstalled.

Too-Great-Risk. The action the system took was successful but it later realized that under the circumstances in which it was taken there was a fair chance it would have had a terrible outcome.

Example: An important file is e-mailed to someone, but the system later learns that they were at a location that receives large e-mails unreliably, and hence the file should have been transferred some other way.

Misclassification. Several actions on an object failed in a row, and the system realized that it had classified the object being in one category when in fact it was in another.

Example: Several attempts to write a file on a given storage device fail. The system later realizes that the storage device is write-once removable disk that has already been filled.

6 A Preliminary Collection of Reflective Critics

We have formulated a variety of reflective critics in addition to those listed above. Table 2 lists several more reflective critics expressed informally in English. Note that further mental event predicates are required to express these. This collection is by no means final—formulating new reflective critics is for us an ongoing process.

Table 2. More Reflective Critics

Credit-To-Wrong-Action. Credit was given to a given action for producing an outcome, when it was really produced by another action.

Assumed-False-Preconditions. Actions seem to be failing, and the system realizes that preconditions for those actions that had been assumed in fact did not hold.

Unable-To-Decide. Several methods seem to apply to the current problem, but a decision has failed to be made about which to select.

Wasted-Reasoning. While formulating a plan of action, the system realizes that the situation had changed and the problem had taken care of itself.

Lack-of-Experience. The system has had only a few experiences dealing with this problem.

Ignored-Relevant-Object. The system had expected a particular outcome from a given action, and in fact a different outcome had ensued because it interacted with an object that had previously not been noticed.

Transient-Conditions. The system had been depending on certain conditions to hold for a period of time, but in fact those conditions only held more briefly.

Misremembering. The system's memory of an event was revealed not to have been an accurate description of the original happening.

7 Further questions

While we are on our way towards the catalog of critics that Sussman advocated, there remain many important questions to be explored. Within an agent system, do reflective critics run all of the time like daemons in the background, or are they invoked only under certain circumstances? If the latter, how do we decide when to invoke reflective critics and when to suppress them? Should they respond to errors noticed by ‘action’ critics, as in the above example, or should they be awoken only when certain items have entered recent deliberations? If there are indeed a large number of reflective critics, as we believe, how should they be prioritized and selected between? Do we need *additional* super-reflective layers to reflect on the activities of our reflective critics?

After recognizing a criticism in recent deliberations, how might we respond to it? Finally, what is the nature of reflective debuggers, and the relationship between reflective critics and reflective debuggers? While we did not explore reflective debuggers in this paper, a system is unlikely to be effective if it only sees flaws and not ways to repair or otherwise deal with those flaws.

8 Conclusions

In this paper we presented a preliminary collection of reflective critics that we are exploring with the goal of building more self-reflective and self-repairing agent systems. Such reflective critics are meant for systems that engage in ‘general purpose’ reasoning, where the deliberative layers contain a wide variety of agent types that produce a wide range of inferences, and hence are subject a comparably wide variety of bugs. So while we have presented a broader range of reflective critics than may be needed for systems that engage in only a few kinds of reasoning, for systems that engage in many different kinds of reasoning we expect most of these reflective critics will be useful.

Reflective critics have many potential roles to play within a problem solving system. Critics can comment on different stages of the problem solving process and be applied at different times. They can anticipate errors that could occur, they can complain about problems that are presently occurring, or they can criticize problems whose sources lie in states that existed in the past. A critic that applies in the future helps select and avoid problematic future actions, or repair ideas prior to their selection. A critic that applies at the moment helps modify and tune (or choose to abandon) a method being applied. A critic that applies in the past helps us ‘brood’ over failed methods, learn from and modify those methods, change the conditions under which to invoke them again, and so forth.

We believe that developing a large collection of such reflective critics is essential to the development of more robust agent systems. We are presently working on an implementation of a layered architecture that incorporates these reflective critics as a central component. While this work is still preliminary, we believe it represents a step towards more self-reflective and self-repairing agent systems.

References

1. Minsky, M. (forthcoming, 2003). *The emotion machine*.
2. Sloman, A. (2001). Beyond Shallow Models of Emotion. *Cognitive Processing*, 2(1), 178-198.
3. Sussman, G. J. (1973). *A computational model of skill acquisition* (Phd thesis). Department of Mathematics, MIT.
4. van Harmelan, F., Wielinga, B., Bredweg, B., Schreiber, G., Karbach, W., Reinders, M., Voß, A., Akkermans, H., Bartsch-Sporl, B., and Vinkhuyzen, E. (1992). *Knowledge-Level Reflection*. In B. Le Pape and L. Steels, (Eds.), *Enhancing the Knowledge Engineering Process*, pp. 175-204, Elsevier Science, Amsterdam, the Netherlands.
5. Fox, S. and Leake, D. (1995). Using introspective reasoning to refine indexing. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*.
6. Stroulia, E. and Goel, A. (1995). Functional Representation and Reasoning in Reflective Systems. *Journal of Applied Intelligence*, 9(1), 101-124. Special Issue on Functional Reasoning.
7. Laird, J. E., & Rosenbloom, P.S. (1996). The evolution of the Soar cognitive architecture. In D. M. Steier and T. M. Mitchell, (Eds.), *Mind Matters: A tribute to Allen Newell*, pp. 1-50, Mahwah, NJ: Erlbaum.
8. McCarthy, J. (1995). Making robots conscious of their mental states. *AAAI Spring Symposium on Representing Mental States and Mechanisms*.