

Reflective Introspective Reasoning Through CBR

Susan Eileen Fox
fox@macalester.edu
Macalester College
1600 Grand Avenue
Saint Paul, MN 55105 USA

Abstract

In recent years, “introspective reasoning” systems have been developed to model the ability to reason about one’s own reasoning performance. This research examines “reflective” introspective reasoning: introspecting about the introspective reasoning process, itself. We introduce a reflective introspective reasoning system that uses case-based reasoning (CBR) as its central reasoning method. We examine the advantages of such a system, and attempt to classify the reasoning failures within introspective system that indicate a need to reflect higher.

Introduction

In artificial intelligence, meta-reasoning systems have been used for a variety of purposes: to predict the behavior of other agents, to guide the acquisition and application of domain knowledge, and to “learn” by adjusting the system’s own reasoning processes in response to experience. Meta-reasoning systems model the ability humans have to reason about our own and others’ reasoning performance. Systems that apply meta-knowledge and meta-reasoning to improvement of their own reasoning processes are called “introspective learning” systems.

The ability to reason introspectively requires knowledge of one’s own reasoning methods, and observation, evaluation, and alteration of those methods when needed. A similar ability has been documented in studies of human reasoning behavior, and modeled with a variety of artificial intelligence techniques. Introspective reasoning systems often lack a critical component of human meta-reasoning: the ability to introspect about our introspections themselves. Introspective reasoning is often applied only to reasoning in service of some underlying task, such as navigation planning. An introspective reasoner that can apply its reasoning to its own introspective processes and repeat this reflection upon itself indefinitely is called “reflective” (Ibrahim, 1992).

We have developed a reflective introspective reasoner that learns by altering its reasoning methods at all levels. RILS¹ analyzes both its task-level planning process and its introspective reasoning process, using a unified reasoning method, case-based reasoning (CBR), for all its tasks. Re-using CBR simplifies the introspective model of the system’s reasoning processes and facilitates reflection.

A key issue for a reflective reasoner is deciding when to reflect. RILS uses failure-driven learning; reflective introspection occurs when an impasse is reached.

In the next section, we provide some background on “reflection,” introspective learning in humans, and other introspective reasoning systems. We then describe the RILS approach to reflective introspection, and we classify the kinds of reasoning failures RILS uses to trigger reflective introspective reasoning.

Background

Reflection

The term reflection, as it is used in this paper, refers to systems which can shift the focus of their processing from the current task to the problem-solving task itself, and can repeat this shift of processing indefinitely (Ibrahim, 1992). As a system shifts to a higher-level task, it constructs a “reflective tower” of reasoning processes. Each process analyzes and alters the one beneath it, which is suspended until the higher-level process completes its task.

While not a requirement, Reflection is easiest to achieve if the same reasoning method is used at all reasoning levels. In order to introspect, the system must have a model of its own reasoning processes. The model becomes more complex as the number of different reasoning methods grows. Simplicity in the model enables the system to meaningfully alter more features of its processing, leading to greater flexibility.

The strength of a reflective system is its flexibility: every aspect of the system is open to adaptation and improvement, as the system responds and learns from its experiences. The drawback to a reflective system is the potential of the system to reflect infinitely without making forward progress at any level. The system, when in doubt, must choose not to reflect and continue processing. Many reflective systems only shift attention to a higher level when an explicit failure, particularly a catastrophic failure occurs.

Human Introspective Reasoning Behavior

Several studies have found evidence that humans engage in introspective learning behavior: altering their reasoning strategies as their experience grows.

Chi & Glaser (1980) found differences between the reasoning strategies of experts and novices: experts approach problems in a more playful way, spend more time

¹Reflective Introspective Learning System

analyzing of a problem before attempting to solve it, and understand better the important features of a problem. They suggest these strategies are learned as part of the process of becoming an expert.

Flavell, Friedrichs, & Hoyt (1970) found that older children were better than younger children at monitoring how well they had performed a given task and judging when they had completed it. The development demonstrated here indicates an awareness of one’s own reasoning processes, as well as learning from that awareness to perform better. Kreutzer, Leonard, & Flavell (1975) found that children improved their understanding of their own memory processes as they became older: asked to describe strategies for remembering things, older children tended to describe many strategies and their outcomes, younger children very few.

Kruger & Dunning (1999) found that competence at a task was related to the ability to accurately judge one’s own competence. This suggests that introspective skills are integrally intertwined with particular domain skills.

In all these cases humans show an improved performance on a domain task when they also show evidence of planful, introspective learning behaviors.

Introspective Reasoning Systems

A variety of approaches to introspective reasoning systems exist. A few have examined reflective introspection.

SOAR is a rule-based system which does deliberately address reflection (Rosenbloom, Laird, & Newell, 1993). SOAR’s rule base may contain rules which control the rule selection process, among others. SOAR can learn new behaviors by creating new meta-rules. Its meta-rules cannot affect all portions of its reasoning process.

The Massive Memory Architecture performs both introspective reasoning and case-based reasoning in a task-driven manner (Arcos & Plaza, 1993). Autognostic uses a “Structure-Behavior-Function” model to represent reasoning processes (Stroulia & Goel, 1995). RAPTER uses model-based reasoning: an explicit model of “assertions” describing the ideal reasoning behavior is used to diagnose failures (Freed & Collins, 1994). These systems do not include reflective capabilities.

Meta-AQUA maintains reasoning trace templates (Meta-XP’s) which describe the patterns of reasoning that indicate reasoning failures (Cox, 1996). Meta-AQUA’s Meta-XP’s could be applied to the introspective process itself, but reflection is not the focus of the project.

IULIAN integrates introspective learning with the overall domain task in a way that permits reflection (Oehlmann, Edwards, & Sleeman, 1994). IULIAN uses case-based planning to generate both domain introspective plans, but, as in SOAR, its introspective plans have incomplete access to the mechanisms which use them.

The ROBBIE system (Fox & Leake, 1995), the precursor to RILS, is related to the model-based reasoning systems described above. It contains an explicit collection of assertions which describe the ideal reasoning process of its case-based planner. ROBBIE’s uses its model to perform detection, diagnosis, and repair of reason-

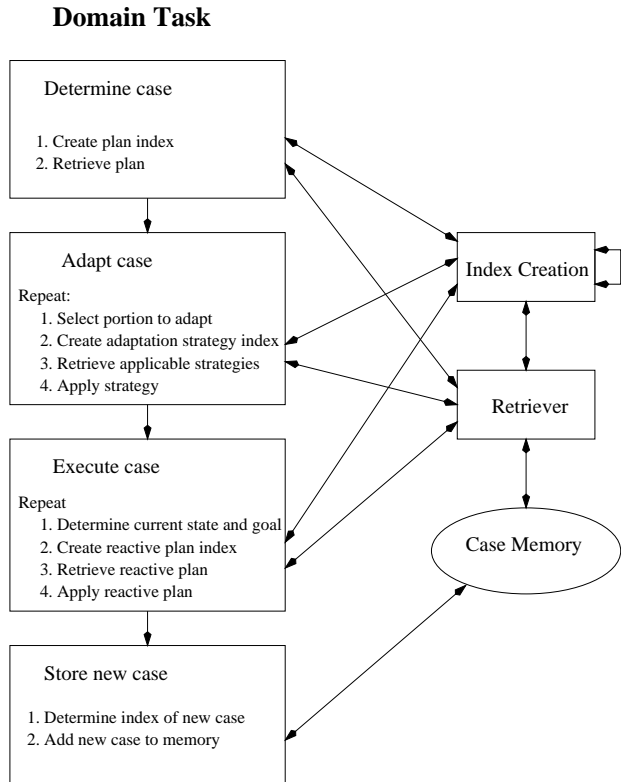


Figure 1: The domain task architecture of RILS: reuse of CBR

ing failures. It is not reflective; its introspective model describes only its underlying planning system. ROBBIE does reuse its CBR processes for multiple planning tasks. RILS was developed from ROBBIE as a *case-based* introspective learner that retains an explicit model of its reasoning embodied in a set of introspective cases.

The Reflective Introspective Learning System

RILS performs route planning for a simulated robot, supported by case-based and introspective learning. Case-based reasoning is the central reasoning method for all RILS tasks. The case memory stores cases for creating and executing route plans, and also stores “assertion cases” that, taken together, comprise its introspective model. The model captures the reasoning processes of both planning and introspective tasks.

RILS’ Domain Task

RILS operates in the same domain as the ROBBIE system (Fox & Leake, 1995); it navigates a simulated robot through a domain of streets, using case-based planning to create high-level plans and case-based reactive planning to interactively execute the plans in its simulated domain. Figure 2 shows a sample of RILS’ domain.

RILS reuses its case-based index creation, retrieval, and case memory for multiple domain tasks: creation of an index, selection of a high-level plan, adaptation of the plan, and selection of reactive planlets to execute

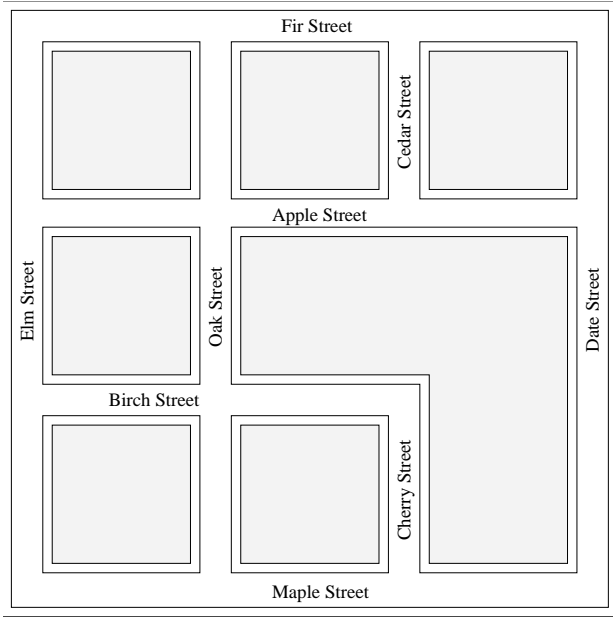


Figure 2: A sample of RILS' domain

its goals. Learning occurs at the domain level through the storage of new high-level plans once they have been successfully executed, and through the addition of index creation rules which are learned through introspective reasoning. The introspective reasoner recognizes and repairs those times when the plan case retrieval retrieves a suboptimal match to a given situation.

The following example demonstrates how RILS' operates at the domain level, and how introspective learning occurs to the domain reasoning. Suppose that RILS has as a new goal to move from the corner of Maple and Elm Streets to the corner of Cherry and Maple Streets. It has in its case memory two potential matches to this problem: a plan from the corner of Maple and Elm to the corner of Birch and Oak, and a plan from the corner of Apple and Elm to the corner of Apple and Oak. Without any learned indexing rules, RILS prefers the first plan to the second one, because it shares the same starting location as its current problem. RILS adapts the selected plan and successfully executes it to arrive at its goal.

The process of executing the plan also streamlines it, so that the plan RILS stores back into its case memory involves merely turning east and moving along Maple to Cherry Street. The storage of new plan cases is the most basic level of learning in RILS.

The introspective reasoning system monitors the domain-level reasoning process, and detects a reasoning failure: the plan being stored into memory is more similar to an unretrieved case (from Apple and Elm to Apple and Oak) than to the retrieved one. This triggers introspective reasoning to diagnose and repair the domain level reasoning. The ultimate repair is to add an indexing rule to detect and prefer case matches where the general direction of movement is the same (i.e., "move straight east"). We discuss the introspective task in more detail in the next section.

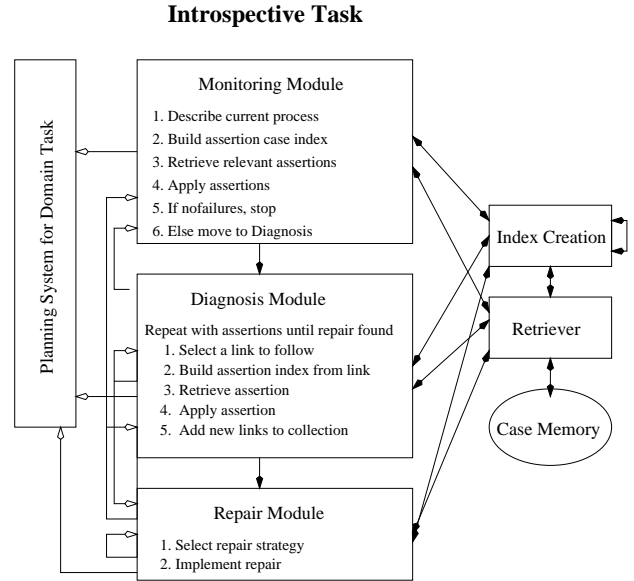


Figure 3: The introspective task architecture of RILS: monitoring applies only to domain-level reasoning; diagnosis and repair apply to both domain and introspective modules.

RILS' Introspective Task

The goal of the introspective reasoning system is to detect reasoning failures in all portions of the system itself and, where possible, to correct the system's processing to avoid repeating the reasoning failure in the future. A "reasoning failure" is any situation which is not predicted by the system itself, whether it is a failure or a success in terms of the domain task. For example, an unanticipated opportunity to achieve a goal is as much of a reasoning failure as a failure to achieve a goal.

In order to reason about its own reasoning, RILS has a model of its ideal reasoning behavior. Figure 1, which shows RILS' domain reasoning, also represents abstractly one portion of the introspective model. Figure 3 shows RILS' introspective reasoning, represents the rest of its introspective model, as well. For each component of the reasoning process, RILS has a collection of "assertions" at multiple levels of abstraction which describe in detail RILS' expectations about its own performance.

RILS actively monitors each step in the domain reasoning to verify that the actual behavior corresponds to its model. When an assertion fails to be true of the system's actual behavior, RILS suspends the lower-level reasoning task and takes over, attempting to diagnose and repair the detected failure. The diagnosis module searches through its model, examining those assertions which are causally related to the detected failure, until it finds a related assertion which has a repair recommended for it. Once a repair is performed, control returns to the planning process.

While RILS may actively monitor its underlying reasoning task, it cannot actively monitor its introspective reasoning process. To do so would lead immediately to an infinite reflective tower: RILS would monitor₁ its

monitoring, then have to monitor₂ the monitoring₁, and so forth. Instead, RILS waits for “impasses” in its introspective process: places where the reasoning process cannot continue, including unexpected catastrophic failures at the domain task level. If RILS discovers explicit evidence that the introspective reasoning process itself is flawed, it suspends the introspective process and reflectively applies its diagnosis and repair processes to the introspective task. These impasses are detailed in the next section.

It seems intuitive that RILS should actively monitor its domain-level reasoning and passively monitor its introspective reasoning. Route planning and execution interact with a highly complex, poorly understood world, where reasoning failures are common. RILS’ understanding of the state of the world at any time is limited. On the other hand, the introspective reasoner has a much more restricted domain: the reasoning of the system itself, and must be assumed to be more reliable.

Each component of RILS’ introspective reasoning is implemented using the same case memory and case retrieval mechanisms as are used in the planning task. The introspective model, rather than being a monolithic collection of assertions, is represented by “assertion” cases stored separately in the case memory. When monitoring the planner’s reasoning, RILS constructs an index representing the current point in the reasoning process, and uses it to retrieve those assertion cases which are applicable to that point. When diagnosing a reasoning failure, RILS retrieves cases based on causal links which are stored in each assertion case. RILS’ repair module retrieves and applies repair plans from the case memory.

Assertion cases must contain sufficient information to retrieve them when needed. Consider the sample assertion case show in figure 4: the assertion case says that the diagnosis module will only consider assertions causally relevant to the current problem. Each assertion case contains an assertion (and information to help apply it), links to other causally-related assertions, applicable repair strategies, and statistics on the assertion’s use and success. Assertion cases are retrieved to support monitoring and diagnosis of reasoning failures. As part of the ROBBIE process, a general vocabulary was developed which describes a wide range of assertions about reasoning processes: the generality of this vocabulary is demonstrated by the ease with which the introspective model was transformed to include a reflective component.

We illustrate the reflective aspect of RILS with the following example, extending the example in the previous section. Suppose that RILS experiences the reasoning failure described above: it selects an inappropriate case, diagnoses the failure, and learns a new indexing rule to avoid the faulty selection in the future. All this requires introspection only about the planner. RILS performs reflective introspection when the introspective reasoner itself is faulty. In this case, suppose that the repair module incorrectly instantiated the new indexing rule, so that it will not be retrieved when it is applicable. Some time in the future, RILS plans a route from the corner of

```
(diagnose-spec2
 (assertion diagnosis specific 2 during)
 (and (contains-part assert-case links)
      (member-of-structure
       (part-value assert-case links)
       checked-assertions))
 (variables assert-case checked-assertions)
 (links (abstr (diagnosis general 2))
        (prev (diagnosis specific 1))
        (next (diagnosis specific 3)))
 (repair)
 (statistics (uses 12)
            (failures 0)))
```

Figure 4: An assertion case for the diagnosis component: “Every assertion retrieved during diagnosis will have a link to one already under consideration”

Birch and Elm Streets to the corner of Birch and Oak. It should apply the new indexing rule, but fails to find it. It therefore retrieves an inappropriate case in exactly the same manner as described above. When the domain-level reasoning failure is discovered and diagnosed, RILS notices that this is exactly the same error as it supposedly corrected earlier. This is the evidence RILS needs in order to invoke reflective introspection. It will suspend the introspective task, and apply the diagnosis and repair modules to the introspective task itself.

Diagnosing and repairing this type of introspective reasoning failure is difficult, because of the long time lag between the actual failure and its detection. RILS keeps a history of its past reasoning decisions in order to be able to diagnose an unbounded distance into its past reasoning. At present this history is used to attempt a diagnosis on introspective reasoning failures: repairing such failures is still work in progress. In the next section we classify the kinds of impasse situations and reasoning failures RILS uses to determine opportunities for reflective introspection.

Reflective reasoning failures in RILS

RILS responds reflectively only to explicit failures which indicate a flaw in the introspective reasoner. Because each module reuses CBR for its reasoning, the sorts of failures RILS must look for are similar for each module. The underlying cause of each failure type differs depending on the module in question: we will examine each module in turn below. The basic categories of failures are:

1. Case memory lacks a required case;
2. RILS fails to retrieve a relevant case;
3. RILS retrieves an irrelevant case; and
4. RILS improperly applies a retrieved case.

Currently, RILS detects these reasoning failures and attempts to reflectively introspect about them. It does not yet repair its introspective process, though work on that aspect is underway.

The Monitoring Module

The monitoring process examines only the route planner. It retrieves those assertions which are currently relevant and checks that no assertion is violated.

If an assertion case is missing from the case memory, that implies that the introspective model is incomplete, and hence inaccurate. This is a troubling failure, because RILS alters its reasoning processes based on the assumption that its introspective model captures the ideal behavior of the system. If a case is missing, then it is possible that RILS would alter itself incorrectly. This is not a failure type that we anticipate RILS handling in any deep manner: It might be able to conjecture that a case is missing and then let a human user/programmer assist in correcting the situation.

Failing to retrieve a relevant case is an easier problem to detect, though detection may be delayed some period from the occurrence of the failure. A failure to retrieve reflects some flaw in the index of the assertion case, or a flawing the index creation and retrieval mechanism. RILS can examine its case memory for cases that are referred to by other assertion cases in memory but that have not been retrieved along with them. RILS keeps statistics on the application of assertion cases to help with this analysis. Correcting this failure requires the alteration of the indexing and retrieval methods for assertion cases; we expect RILS to incorporate this repair in the future.

Because the monitoring module knows the context of the planner at a given moment, recognizing an irrelevant assertion case is easy to detect on the spot: one case in which reflection can occur at the moment of the reasoning failure. As before, this indicates either a flaw in the index of the particular case retrieved, which is easy to check and correct, or a flaw in how the system creates indices and retrieves assertion cases. RILS can alter the indexing rules for plan cases; we should be able to extend this to altering the indexing of assertion cases as well.

A misapplication of a retrieved case could be detected if the lack of it leads to an unexpected catastrophic failure of some sort. RILS examines catastrophic failures at the route planning level to determine if the monitoring process failed to detect a problem before the catastrophic failure occurred, and considers the possibility that an assertion was misapplied.

The Diagnosis Module

The diagnosis module retrieves assertion cases and evaluates them in much the same way as the monitoring module. Therefore, many of the comments made about monitoring also apply to diagnosis. The main difference is in how the diagnosis module chooses which cases to retrieve: it starts with an assertion case which is known to be a failure, and then performs a heuristic-guided breadth-first search through those assertions which are causally related to the detected failure. It uses the causal links each assertion contains.

For reflective diagnosis, it may be initially unclear which assertion has failed, RILS creates a set of potential failed assertions and searches in parallel starting from

each possibility.

A missing assertion case is just as much of a problem for diagnosis as for monitoring. RILS could distinguish between diagnosis and monitoring by which module was most recently in use, but would have just as much difficulty recognizing the lack and determining a repair.

Failing to retrieve a relevant case for the diagnosis module could be due to two different failures. Like the monitoring module, diagnosis could miss a case due to indexing problems. Because diagnosis is performing highly targeted retrievals, however, this is a problem likely to be discovered as it happens. Alternatively, if assertion A should contain a link to a causally related assertion B, but lacks that link, then B could be overlooked. This problem must be detected by examining the cases in memory and their usage statistics. Retrieving an irrelevant case is, again, easy to detect.

Circumstances that indicate a problem with the diagnosis module include times when the diagnosis process fails to find any applicable repair. Alternatively, the diagnosis module may attempt to evaluate an assertion whose value depends on one that was overlooked or misapplied, and may be unable to complete its evaluation.

The Repair Module

The repair component retrieves and uses repair cases which describe how to change the system to correct a diagnosed reasoning failure.

A missing repair case is, as for assertion cases, difficult to detect, unless a repair is referred to in an assertion case and does not exist in the case memory.

Failing to retrieve a relevant repair or retrieving an irrelevant one may be detected immediately, as the repair module performs very targeted case retrieval. It would be repaired, as above, by either altering the indexing of the repair case, or altering the index and retrieval methods of the system.

Misapplying a repair strategy is a difficult failure to detect. A repair could be executed, and might not correct the experienced failure. As in the example in the previous section, RILS may detect this type of failure if it faces a similar situation again and generates an identical repair.

Conclusions

RILS demonstrates the power of case-based reasoning to serve as the central reasoning method of a system performing a wide range of reasoning tasks. By having one approach to reasoning, RILS has a fairly simple, powerful model of its reasoning processes. It uses that model to introspect about its reasoning process: learning from failures of its domain-level reasoning, detecting opportunities to reflectively introspect and, eventually, learn by improving its introspective reasoning processes.

A system that combines task reasoning with a reflective introspection capability should be able to respond with flexibility to a complex environment, by re-tooling its knowledge and processes at all levels of abstraction. This seems to be a talent which humans possess, as we

improve our reasoning strategies with experience. RILS provides one possible abstract model of this process.

Reflection permits a system to adapt itself to its surroundings, but poses a hazard if left unchecked. We have demonstrated here a “failure-driven” approach to controlling reflection: only when clear evidence exists that the introspective reasoning is flawed will RILS choose to reflect. Our future work on RILS will complete the reflective skills it has by giving it the tools to repair, not just diagnose, its introspective reasoning processes,

References

- Arcos, J. & Plaza, E. (1993). A reflective architecture for integrated memory-based learning and reasoning. In Wess, S., Altoff, K., & Richter, M. (Eds.), *Topics in Case-Based Reasoning*. Springer-Verlag, Kaiserslautern, Germany.
- Chi, M. & Glaser, R. (1980). The measurement of expertise: a development of knowledge and skill as a basis for assessing achievement. In Baker, E. & Quellmalz, E. (Eds.), *Educational testing and evaluation: Design, analysis and policy*. Sage Publications, Beverly Hill, CA.
- Cox, M. (1996). *Introspective multistrategy learning: Constructing a learning strategy under reasoning failure*. Ph.D. thesis, College of Computing, Georgia Institute of Technology. Technical Report GIT-CC-96-06.
- Flavell, J., Friedrichs, A., & Hoyt, J. (1970). Developmental changes in memorization processes. *Cognitive Psychology*, 1, 324–340.
- Fox, S. & Leake, D. (1995). An introspective reasoning method for index refinement. In *Proceedings of 14th international Joint Conference on Artificial Intelligence*. IJCAI.
- Freed, M. & Collins, G. (1994). Adapting routines to improve task coordination. In *Proceedings of the 1994 Conference on AI Planning Systems*, pp. 255–259.
- Ibrahim, M. (1992). Reflection in object-oriented programming. *International Journal on Artificial Intelligence Tools*, 1(1), 117–136.
- Kreutzer, M., Leonard, M., & Flavell, J. (1975). An interview study of children’s knowledge about memory. *Monographs of the Society for Research in Child Development*, 40. (1, Serial No. 159).
- Kruger, J. & Dunning, D. (1999). Unskilled and unaware of it: how difficulties in recognizing one’s own incompetence lead to inflated self-assessments. *Journal of Personality and Social Psychology*, 77 (6).
- Oehlmann, R., Edwards, P., & Sleeman, D. (1994). Changing the viewpoint: re-indexing by introspective questioning. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, pp. 675–680. Lawrence Erlbaum Associates.
- Rosenbloom, P., Laird, J., & Newell, A. (1993). *Meta Levels in Soar*, Vol. I, chap. 26. The MIT Press.
- Stroulia, E. & Goel, A. (1995). Functional representation and reasoning in reflective systems. *Applied Artificial Intelligence: An International Journal, Special Issue on Functional Reasoning*, 9(1), 101–124.