Flexible Integration of Multiple Learning Methods into a Problem Solving Architecture

Enric Plaza Josep Lluís Arcos
Institut d'Investigació en Intel·ligència Artificial, IIIA-C.S.I.C.
Camí de Santa Bàrbara, 17300 Blanes, Catalunya, Spain.
{plaza | arcos}@ceab.es

Abstract. One of the key issues in so-called multi-strategy learning systems is the degree of freedom and flexibility with which different learning and inference components can be combined. Most of multi-strategy systems only support fixed, tailored integration of the different modules for a specific domain of problems. We will report here our current research on the Massive Memory Architecture (MMA), an attempt to provide a uniform representation framework for inference and learning components supporting flexible, multiple combination of these components. Rather than a specific combination of learning methods, we are interested in an architecture adaptable to different domains where multiple learning *strategies* (combinations of learning methods) can be programmed or even learned.

1 Introduction

The first issue we have to debate is the relationship of inference and learning, and the second is the kind of representation we use to support an integration of inference with multiple learning methods. Regarding the first issue, the integration of learning methods, we agree with the Inferential Theory of Learning (ITL) approaches such as [2] and [10] in that learning methods are a special kind of inference methods. Our approach differs from ILT, however, on the characterization of the inference components that made up learning methods. For ILT approaches, the components that made up learning are inference operators like deduction, analogy, generalizations, etc. In our approach, learning is a kind of metalevel inference, that is to say, a kind of inference that requires a certain kind knowledge about the system behavior and state itself, e.g. about when a failure occurs, how learning can overcome it, etc [6]. The issue of the self-model is crucial because it formally defines what a learning method has to know so as to be able to learn. That is, a self model specifies the relationship between a learning method in a systems and the system as a whole: it specifies what knowledge the learning method can effectively know about the system in which it is integrated. There are sorts of integration, like the PRODIGY system, where learning methods are external modules that each has a clear and specialized model of the problem solver [1]. In the THEO architecture [3] several learning methods are integrated but the self-model used are not clear. Our approach is closer to THEO, since the MMA is implemented as a reflective language [5], but MMA has been designed having in mind the reflective nature of learning from problem solving and with the purpose of investigating in clear self-models required for learning.

The second issue to discuss is the nature of inference in the MMA, since we do not agree with the ILT components. Our approach derives from the knowledge-level analysis of expert systems and the so-called conceptual frameworks developed for the design and construction of KBSs. These conceptual frameworks for knowledge modelling like KADS [11], components of expertise [10] are based on the task/method decomposition principle and the analysis of knowledge requirements for methods.

2 The Massive Memory Architecture

The MMA is an experimental framework for experience-based learning and reasoning. Every episode of problem solving of MMA is represented and stored as an episode in memory. This is the main point of the reification process: create the objects that can be usable for learning and improving future behavior. MMA records memories of successes and failures of using methods for solving tasks. An episode involves not only a global task (e.g. diagnosis) but also all its subtasks and the methods that achieve them are stored, allowing learning methods to be used at different levels of granularity and from multiple examples. Our hypothesis is that the problem solving process in MMA can be described into a collection of abstract inference components. Those components are *tasks* (or goals), *methods* (or ways of achieving a goal) and *theories* (sets of methods adequate for some class of objects). This means that these elements constitute the self-model of the system and have to provide the information that will be accessed by introspection by the learning methods.

A task is engaged when exists a query to the system. One task can be achieved in some different ways. A method is a specific way to achieve one task. A method is an evaluable object that is recursively decomposed into subtasks (queries to other objects), until some direct methods are executed. This recursive decomposition of task into subtasks is called the task/method decomposition and is common to all KBS conceptual frameworks [10]. The domain knowledge is organized around objects called *theories*. For instance, the set of methods applicable to persons can be defined in a theory of `person`, stating the available methods for each task to be solved about persons. Methods to solve a specific task are grouped in an object called metafunction. A metafunction holds a set of methods and preferences to choose among them. In addition, when a method fails to achieve the task, a metafunction allows backtracking to other not failed methods.

The MMA approach is uniform and this entails that the problem solving *process* is also described in the system in terms of tasks and methods. For instance, if there is no method specified for solving a given task, the *task* of the problem solving *process* is to find such a method; or if there are more than one method that can possibly solve a task, a *task* of problem solving *process* is to choose among them. A way to do it is trying them out until one works: that would be a default *method* for such a *task*. This approach involves backtracking as a constitutive aspect of MMA, as can be expected for plausible reasoning applications. One important thing is that any time some knowledge is required by a problem solving method, and that knowledge is not *directly available* there is an opportunity for learning. We call those opportunities *impasses*, following SOAR terminology [4], and the integration of learning methods is realized by methods that solve these impasses. This type of methods are called *inference methods.*

An inference method is a type of method with a self-model of the architecture and at least a retrieve/select subtask decomposition. The goal of *retrieve* task is to obtain a set of plausible useful past episodes from the memory. MMA provides a set of basic retrieval methods that can be combined to build more complex retrieval methods. The goal of *select* task is to rank the precedents in some criteria to work first with the most plausible past episodes. MMA provides also a set of basic preference methods that can be combined to build more complex preference methods. An inference method combines the precedents in some programmable way and, finally, the result is installed in the current task. In this way we can program different learning methods as a different inference methods. Inference methods are organized around theories called *inference theories.* This means that we can also program different strategies to combining the learning methods.

3 Integration of Multiple Learning Methods

In this section we want to show how different strategies of learning can be developed and integrated into the Massive Memory Architecture. We will present two families of learning methods: Analogical methods and Inductive methods and finally we will discuss the combination of them.

Analogical methods are a family of inference methods that follow a Retrieve/Select task decomposition. The characteristic of analogical methods is that the `Retrieve` method uses a similarity-based methods. `Select` methods can also be based on similarity or can be domain-specific, knowledge-intensive methods. The most basic (less domain-dependent) method for analogy uses a retrieval method that finds all objects having some successful method solving the task at hand, and no preference knowledge is provided so a random selection is performed. This method is the most general, less focused, less informed, and less domain-dependent analogical method. If we have some knowledge of the domain we can add it and focus the analogical reasoning. Adding this knowledge in the retrieval or/and the preference methods we can define some more specific, i.e. domain specific, analogy methods. A form of useful domain knowledge are determinations (they are a form of functional dependence, as in "Spoken language of a person depends on his/her nationality" [9]). Determinations in MMA can be used during Retrieve or Select tasks of analogy. During retrieval MMA has retrieve methods that find from memory those cases or episodes that comply to the determination (e.g. in solving the spoken-language task, retrieve only those examples with the same nationality as the current problem). During selection tasks MMA has a preference method that prioritizes , from the retrieved cases, those complying to the determination. Thus different analogical and case-based methods can be included into MMA in a uniform way and only depending on the knowledge available for a domain, as is essential for KBS conceptual frameworks. Since to all tasks and subtasks can be ascribed a specific method MMA is capable of handling multiple methods for different subtasks? In fact multiple methods for a single task can be applied, as we'll show presently with inductive and analogical methods,

Inductive methods are another family of inference methods that follow a Retrieve/Construct task decomposition. `Retrieve` methods obtain a set of past solved examples from which to learn the knowledge to solve a particular task. `Construct` methods are domain-specific methods that compare the examples and build a domain theory for that specific task. We can build an inductive method that learns to solve all the problems that an analogical method solves case by case. This method retrieves the same examples as the analogical method and constructs a metafunction with all the methods successfully used in those past examples. In the language-nationality example, the inductive method learns the function associating nationalities with languages. That is to say it learns in one step the domain knowledge needed by the method that solves the spoken-language task, while the analogical method solves the task acquiring the needed knowledge on a case-by-case manner.

Another possibility in MMA is to combine for the same task an inductive method and an analogical method. In this situation the inductive method acquires not all domain knowledge but only those methods frequently used, resulting in a domain theory that is efficient although incomplete. The task now has available two methods: the induced method and the analogical method and a preference from the first to the second. In this strategy first the general knowledge is used and if it fails then the analogical method tries the specific cases retrieved from memory. this strategy embodies the general form of an imperfect theory (the induced knowledge) plus a set of exceptions or unusual cases out of the scope of the theory and for being exploited by an analogy method.

4 Conclusions and Future Work

MMA provides a flexible framework for integration and experimentation of the ranges of applicability and/or utility of learning methods. Different analogical, case-based and inductive methods can be included into MMA in a uniform way and only depending on the knowledge available for a domain. In fact, this is essential idea of KBS conceptual frameworks that we are using for integrating multiple learning methods. Since to all tasks and subtasks can be ascribed a specific method MMA is capable of handling multiple methods for different subtasks and even multiple methods for a single task. Since all decisions, successes, and failures, are stored for every subtask, learning can be applied to any subtask of a global task.

In order to achieve a more autonomous learning capability the study of learning goals [7, 8] is in this context a very important future research line for ML. What we presented here seems however to be an unavoidable previous step because of the necessity of achieving integration and uniform representation of learning methods and problem solving methods in order to have a common ground of comparison. In the next phase of our project (1994-1996) the goal is to enlarge the self-model of the system in such a way that compilative learning methods and other inductive methods are integrated.

Acknowledgements

The research reported on this paper has been developed at the IIIA inside the Massive Memory Project funded by CICYT grants TIC 801/90 and TIC 122-93.

References

[1] Carbonell, J. G., Knoblock, C. A., Minton, S. (1991), Prodigy: An integrated architecture for planning and learning. In K van Lehn (Ed.), *Architectures for Intelligence.* Lawrence Erlbaum Ass.: Hillsdale, NJ.

[2] Michalski, R., (1993), Inferencetial theory of learning as a conceptual basis for multistrategy learning, *Machine Learning*, 11(2-3), 111-152.

[3] Mitchell, T.M., Allen, J., Chalasani, P., Cheng, J., Etzioni, O., Ringuette, M., Schlimmer, J. C. Theo: a framework for self-improving systems. In K Van Lenhn (Ed.) *Architectures for Intelligence.* Laurence Erlbaum, 1991.

[4] A Newell (1990), *Unified Theories of Cognition.* Cambridge MA: Harvard University Press

[5] Plaza, E (1992), Reflection for analogy: Inference-level reflection in an architecture for analogical reasoning. *Proc. IMSA'92 Workshop on Reflection and Metalevel Architectures,* Tokyo, November 1992, p. 166-171.

[6] Plaza, E. Arcos J. L., Reflection and Analogy in Memory-based Learning, *Proc. Multistrategy Learning Workshop.*, 1993. p. 42-49.

[7] Plaza, E., Aamodt, A., Ram, A., van de Velde, W., van Someren, M. (1993), Integrated learning architectures. In P.V. Brazdil (Ed.) *Machine Learning: ECML-93.*, pp. 429-441. Lecture Notes in Artificial Intelligence 667, Springer-Verlag.

[8] Ram, A, Cox, M T, Narayanan, S. (1992), An architecture for integrated introspective learning. *Proc. ML'92 Workshop on Computational Architectures for Machine Learning and Knowledge Acquisition.*

[9] Russell S (1990), *The Use of Knowledge in Analogy and Induction.* Morgan Kaufmann.

[10] Steels, L. The Components of Expertise, *AI Magazine*, Summer 1991.

[10] Tecuci, G (1993), Plausible justification trees: A framework for deep and dynamic integration of learning strategies. *Machine Learning*, 11(2-3), 237-261.

[11] Wielinga, B, Schreiber, A, Breuker, J (1992), KADS: A modelling approach to knowledge engineering. *Knowledge Acquisition* 4(1).