

An Architecture and Language for the Integrated Learning of Demonstrations

Mark Burstein, Marshall Brinn, Mike Cox, Talib Hussain, Robert Laddaga,
Drew McDermott¹, David McDonald, Ray Tomlinson

All at BBN Technologies except ¹ Yale University
10 Moulton Street, Cambridge, MA 02138
burstein, mbrinn, thussain, rladdaga, dmcdonald and rtomlinson @bbn.com
mcdermott@cs.yale.edu

Abstract

POIROT is an integration framework and reasoning control system that combines the products of a variety of machine learning mechanisms in order to learn and perform complex web services workflows, given a single demonstration example. POIROT's extensible multi-strategy learning approach to developing workflow knowledge is organized around a central hypothesis blackboard and representation language for sharing proposed task model generalizations. It learns hierarchical task models from semantic traces of user-generated service transaction sequences. POIROT's learners or *hypothesis formers* develop, as testable hypotheses, generalizations of these workflow traces by inferring task order dependencies, user goals, and the decision criteria for selecting or prioritizing subtasks and service parameters. *Hypothesis evaluators*, guided by POIROT's meta-control component, plan and execute experiments to confirm or disconfirm hypotheses extracted from these learning products. Hypotheses and analyses of hypotheses are represented on the blackboard in the language LTML, which builds on both OWL-S and PDDL.

Introduction

POIROT (Plan Order Induction by Reasoning from One Trial) is an architecture, currently under development and testing, for controlling a multi-step and multi-strategy learning process. The DARPA Integrated Learning Program is exploring techniques for applying a variety of learning and reasoning strategies to enable systems to learn hierarchical procedures from one or at most several examples. In general, this requires the regulated application of inductive, abductive and explanation-guided generalization techniques, and ultimately an ability to do self-guided exploration of the space of activities to confirm or enhance confidence in one's ability to perform the task.

POIROT learns hierarchical task models given a single demonstration of a task, a sequence of semantically annotated web service calls represented using OWL-S (Martin et al, 2007). Its learning is emergent from the work

of a number of independent reasoning components accumulating shared questions, hypotheses and answers on a central blackboard. We believe that in order to make the problem tractable, distinct reasoning processes must be strategically applied to different aspects of the task. POIROT first applies multiple techniques to propose generalizations for aspects of a demonstrated workflow process, and then experimentally tests its own learning. Our approach utilizes a reactive planner as a meta-control executive and a semantic blackboard to manage the hypotheses developed by our largely independent learning components.

The system is guided in this process by *learning goals*, which are managed by a meta-control system called the *learning moderator*. Though the selection of tasks in the system's internal processes are directed by the learning moderator, learning goals may be generated by the learning components themselves, when they cannot resolve issues. POIROT's learning moderator maintains these goals or open learning questions, and triggers components with the declared capability to address them, using techniques commonly found in multi-agent architectures. In addition to managing a number of learning systems, the moderator also manages the allocation and timing of internal tasks associated with such things as planning how to experimentally resolve conflicts between hypotheses created by those different learners, or diagnose and repair problems with the learned task models.

In this paper, we describe POIROT's approach to the integration of multiple learning components, reflective reasoners and planning and experimentation elements. We first describe POIROT's task, its architecture, and the different modules being integrated. Next, we describe key motivations for and design aspects of the Learnable Task Modeling Language (LTML), the interlingua of POIROT. Although modules in POIROT use very different approaches to generate learning hypotheses, their final hypotheses are represented in LTML so they can be compared, and evaluated or subjected to experiments. Finally, we provide a brief summary of previous work in the area, and some preliminary results and conclusions.

Overview of Learning Task

POIROT's objective is to form a generalized hierarchical task model from 'observed' semantic traces of sequences of web service transactions. The domain we are currently using for our investigation is medical evacuation airlift planning, and our model is analogous to the use of web services for trip planning, where one uses different web sites to book airline tickets, reserves local transportation (e.g., taxis, trains buses) and hotel rooms. Differences from these commercial procedures when planning medical evacuations include the ability to reserve aircraft for flights outside normal schedules, the use of helicopters and ambulances rather than trains and taxis, and the reservation of hospital beds rather than hotel rooms. A typical demonstration used as the basis for POIROT's learning shows a number of patients being scheduled to be moved to suitable hospitals, and may also deal with some of the special cases that crop up, such as needing to make sure the patient is accompanied by appropriate medical personnel and equipment.

POIROT's observations are of a sequence of calls to web services that perform tasks such as looking up airports by geographic location, finding available flights to and from those airports, reserving seats on flights and reserving hospital beds at the destinations. Figure 1 shows the GUI used by people to call these services and thereby demonstrate the task. A demonstration given to the human subjects we are comparing our performance to shows how each service is selected, its parameters filled in, and the request submitted.

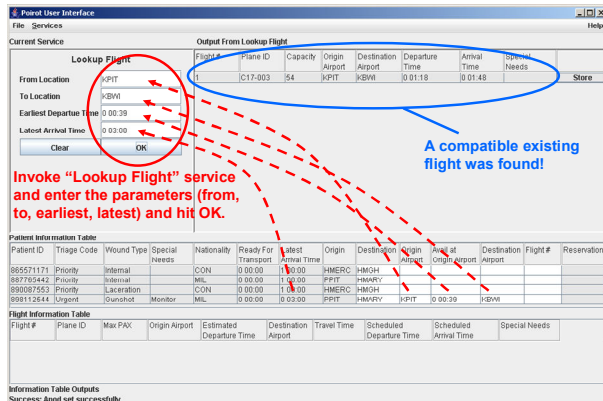


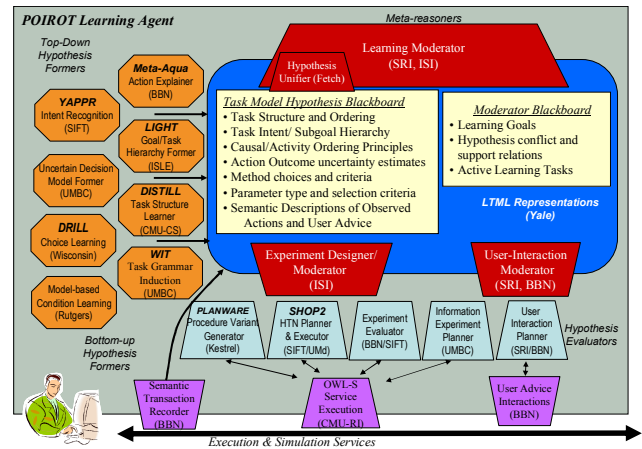
Figure 1: POIROT GUI

Integration by Multi-agent Blackboard Architecture

POIROT's software framework uses a combination of agent and semantic web service architectural elements to integrate multiple learning and execution components as shown in Figure 2. The central hypothesis blackboard where learning hypotheses and commentary are exchanged is built on a persistent RDF store (Sesame), extended to support context-bounded queries and a publish-subscribe interface that is used to wake up POIROT components.

Hypothesis formers use bottom up (inductive) and top down (abductive or explanation-based) generalization techniques to produce hypotheses that are posted on the hypothesis blackboard. The learning moderator communicates about internal system tasks and learning goals with the other modules using a simple handshake protocol, and receives task status feedback through the same blackboard, but using a slightly different API that permits updates to transient facts about status and more control over component activation than the pub-sub mechanism used by other modules.

Figure 2: POIROT Architecture



Learning and Reasoning Technologies

In this section we describe a subset of the reasoning approaches that POIROT will ultimately integrate, focusing on ones that we are currently using during this phase of the research. Our overall learning framework is based in part on the concept of goal-driven learning, described previously (Hunter 1989, 1990, Ram & Hunter, 1992, Cox & Ram, 1994, desJardins, 1995) as 'knowledge acquisition planning' or 'planning to learn'. Goal-driven learning leads to targeted searches for explanations when observations don't fit into the developing model, and to the creation of new knowledge goals about the availability of choices and decision processes associated with task elaboration. POIROT also uses more traditional bottom up approaches that generalize the given observations, using some background knowledge to guide the degree of generalization. We describe the latter first, and then discuss the role of explanation-based method in forming hierarchical task models.

Task Structure Determination

In this, the first year of the project, we have relied primarily on two learning systems for basic task structure determination, each directly generalizing on the observed trace. Both techniques utilize the dependencies between information gathering actions (data querying services) and information consuming ones (when service parameters are

filled in) as a means of inferring dependencies between actions (service calls).

WIT (from UMBC) is based on their prior work on context-free grammar induction (Oates et al, 2002; Yaman and Oates 2007). The demonstration POIROT observes is treated as containing a small set of examples of service call sequences, each handling (that is, scheduling for transit) one of the specified patients. WIT uses these as its set of positive training examples to produce essentially a finite state model of the transitions between steps. While this model can serve as a recognizer for acceptable workflows, it cannot by itself be used to plan one, as it does not model the choices that must be made at branch points.

A second hypothesis former utilizes and extends the work of Winner and Veloso (2002, 2003). This system, DISTILL, automatically learns problem-specific universal procedures from demonstrated examples. DISTILL produces workflow methods that loop over loosely ordered steps that are runtime conditionalized. It develops these conditions by a causal analysis of the actions in the demonstration. Both WIT and DISTILL post their solutions as LTML representations of hypotheses about suitable subtask methods generalized from the trace.

Hierarchical Structure Determination

The LIGHT Hypothesis Former builds on prior work by Langley and his colleagues on the problem of learning of hierarchical task networks from traces of successful problem-solving (e.g., Choi & Langley, 2005) or from observations. The method learns hierarchical representations of planning knowledge similar to that in Nau et al.'s (1999) SHOP2 planning system. It builds sets of methods that together can be used to accomplish the demonstrated task. Methods are encoded with a head (goal state), an ordered set of subtasks, and a set of conditions under which the method should be considered. LIGHT's learning method determines the hierarchical problem decomposition by back-chaining through the activity and observation sequence (a semilattice in which each subplan has a single root node) to form a hierarchical structure that suggests a general method for each subgoal. It indexes methods for subgoals by the goal literals they achieve. Where appropriate, the resulting structures include disjunctive and recursive methods, substantially reducing the time required to solve novel problems, even for problems larger than the training example.

POIROT uses LIGHT as a means of identifying subtasks that be separated out from the overall activity sequence represented by the demonstration trace.

Explanation-Based Composition of Activities

POIROT can be viewed in part as processing a stream of observations to explain how they form a coherent whole. While bottom-up hypotheses formers identify regularities in a demonstrated workflow, aspects of the workflow that are not repeated and do not have observable causal support are more difficult to generalize in a useful way without

applying additional background knowledge to explain their relevance. Explanations are produced by inferring relationships between steps using explanatory schemas that tie the steps to the domain goals of the demonstrator. POIROT's Explanation of Intent Hypothesis Former is based on Meta-AQUA (Cox, 1996; Cox & Ram, 1999), a goal-driven explanation and learning system that has the top level goal to understand each observation in its input stream. It relies on domain knowledge, a case library of prior plan schemas and a set of general explanation patterns that are used to characterize useful explanations involving that background knowledge.

POIROT relies on Meta-Aqua to explain steps that do not have strong observable connection to other steps in the demonstration. For example, steps that gather information used to confirm a precondition for another step, when that precondition is not otherwise known to be required to perform the step in general. If the rationale for the precondition can be plausibly explained by analogy to a historical example and/or by reference to the user's goals, then that explanation justifies the insertion of an ordering constraint into the temporal network of steps represented in the generalized workflow for that context.

Meta-Aqua is also used to associate high level demonstrator goals with observed demonstrator actions. This, independent of local causal relations, helps sets of intentionally related steps to be organized into loosely coupled hierarchical methods.

Using ILP to learn loop ordering criteria

The class of tasks POIROT learns is intrinsically repetitive. Many patients must be scheduled, using limited resources. This leads to choices about priorities that appear in the workflow as choices about such things as the order in which patients are considered for transit. Critical patients are given first crack at being scheduled so that they arrive at their destination in the shortest period of time.

When POIROT recognizes that the procedure involves satisfying a repetitive goal – scheduling a set of patients for transit, and learns a procedure for scheduling one patient, it is ready to address the learning goals associated with learning repetitive, resource-bounded processes. One of these is the question of how (or whether) to impose a particular order on the satisfaction of the conjuncts of this top level goal. While this could be based on an attempt to explain the priorities of the demonstrator, in general there won't be enough information available to do this directly.

This issue has been addressed by Jude Shavlik's group at the University of Wisconsin, who have developed a Hypothesis Formation module (DRILL) using inductive logic programming techniques based on (Goadlich et al, 2004) to learn logical predicates describing the relative ordering relationships among elements of the set of goal objects (i.e., the patients) that aligns with the order in which the those objects were observed to be processed (patients scheduled), and is inconsistent with other orderings. This module will, in general, be triggered when

the meta-control system identifies that there is postulated loop over a set of unordered objects, and it is able to gather as examples the observed order the objects were considered in during the demonstration.

The Learnable Task Modeling Language

The modularity of our architecture depends on an ability to share hypotheses and conclusions using descriptions on the hypothesis blackboard. A group led by Drew McDermott, Mark Burstein and Robert Goldman has developed a language for representing the accumulated hypotheses about generalizations of the observed task. called the Learnable Task Modeling Language (LTML) LTML is based in part on the PDDL planning language (McDermott et al, 2000), and OWL-S, but also includes a simple s-expression syntax for RDF descriptions, and the whole language fully translates into OWL/RDF for storage on the POIROT blackboard. LTML is used to represent planning methods, plans, observations or traces, and domain concepts. It is also used to capture hypothesis annotation information, including such as things as provenance (which learning system proposed it) conflicts between hypotheses, explanations and learning goals.

A central issue in an integrated learning system is the representation of hypotheses. As our focus is on learning process workflows, hypotheses must describe the intentions and procedures learned from the human demonstrator. Workflow methods describe such things as the information passed from one action (step) to another, state effects, action conditions, resources (informational resources, permissions, etc.), and ultimately, certainty of conditional outcomes and measures of utility of methods for achieving goals.

Our current draft of LTML satisfies a number of sometimes competing requirements. It had to be capable of describing hierarchical task methods in a way that was acceptable to HTN planners such as SHOP2. It had to do so in a way that allowed arbitrary fragments of such descriptions to be independently derived or referenced, so that they could be annotated as hypotheses. It had to support descriptions of relationships among hypotheses, such as conflicts or congruences, and support capturing of explanations or justifications for hypotheses. It also had to support internal learning goals and task descriptions that made reference to hypotheses as objects of the discourse.

Thus, for learning and hypothesis formation, LTML needed to enable modular chunks of information to be added, annotated and contradicted by different reasoners. It must be easy to keep track of a family of related but competing hypotheses, and to circumscribe a set of consistent hypotheses so that they may be tested together by construction of an experiment. In contrast, many planners represent plans and actions as single compound expressions which lack these modularity properties. For example, the SHOP hierarchical planner assumes a *plan library* of *methods*, each of which is written in a special-purpose notation specifying steps and their orderings. Our

need for referential modularity required breaking down these task models into sets of assertions internally, and we were able to do this adequately using OWL description logic formalisms, with some additional devices for circumscribing contexts and referencing portions of named descriptions. Figure 3 shows a sample method definition. Note that includes a step referencing the example from which it was generalized.

```
(Method WIT@Method0
  (body
    (seq seq0 (links v0 - Set)
      (acts
        (perform step0
          (med@lookupRequirements)
          (put (med@lur_out => v0))
          (occurrence TraceElt1))
        (loop step1
          (links (v1 (over v0)))
          (body
            (perform step2 (Method1 (in <= v1))))))))))
```

Figure 3: Small LTML fragment

Hypotheses and Experimentation

POIROT also includes an Experimental Design module (CMAX) being developed by Paul Cohen and Clay Morrison at USC/ISI. This module will identify and plan how to explore unresolved causal constraint questions in its target domain of learning. The causal hypothesis generator in CMAX explores which likely causes are simultaneous or otherwise coincidental changes of state; for example, it considers when knowledge revealed by one event is utilized (directly or indirectly using the results of further inference) in a later task; as well as extant but general causal models.

The causal hypothesis evaluator plans simple confirmatory and disconfirmatory experiments to refine the scope of such hypotheses. Causal conditions and other task structure hypotheses are tested by the execution of experiments that systematically vary the demonstration trace to establish the conditions under which an action to test the hypotheses can be performed. Typically this will involve manipulating conditions in the simulated environment provided by the set of available web services.

This hypothesize and test approach provides a domain grounded means to compare and contrast disparate hypotheses about task methods. The results of these experiments can be used to select one hypotheses over another, or to produce new hypotheses that refine the models suggested by others.

Using an HTN Planner for Experimentation

POIROT tests what it has learned by selecting sets of task achievement methods that together should be sufficient to enable an HTN Planner to reenact the demonstration or to perform other suitably similar tasks. The SHOP2 Planner has been incorporated into POIROT by Robert Goldman at SIFT Technologies and Dana Nau at the University of Maryland. It interprets LTML descriptions of HTN

planning domains and problems, develops executable task plans, and executes them using a runtime module called SHOPPER. SHOPPER in turn invokes domain web services through calls to the OWL-S virtual machine (OVM), developed by Katia Sycara's group at CMU Robotics Institute (Paolucci et al, 2003).

This experimental execution module enables POIROT to develop and execute plans to evaluate its shared task models, and provide feedback on the quality of those execution results. The use of semantic web services described using OWL-S both provides the learners with declarative models of the primitive actions of the domain being observed, and enables the system (through the OVM) to perform experiments by actually calling the demonstrated services, even though they were not known to the system beforehand.

Related Work

Investigation of multi-strategy reasoning and learning systems can be traced as far back as to the Pandemonium system (Selfridge, 1959) that combined multiple competing pattern recognition methods. Similarly the area of multi-strategy learning (Michalski & Tecuci, 1994) seeks to organize and taxonomize the various learning methods and to develop procedures whereby multiple learning algorithms can be effectively combined. Michalski's (1994) Inferential Theory of Learning provides a fundamental taxonomy of learning methods that includes inductive, deductive, and analogical algorithms. Furthermore the general idea of a learning goal was established here. In Michalski's view the default learning goal is to increase the total knowledge of the system, though he did not refine the notion.

A large body of work has focused on acquiring and using domain knowledge to reduce planning search. Some of the most commonly used forms of learning for planning include control rules, e.g., (Minton, 1988); case-based and analogical reasoning, e.g., (Kambhampati & Hendler, 1992; Veloso, 1992); and hierarchical and skeletal planning (Knoblock, 1994; Korf, 1985). Many of these methods suffer from the *utility* problem, in which learning more information can actually be counterproductive because of difficulty with storage and management of the information and with determining which information to use to solve a particular problem.

Other work has focused on analyzing example plans to reveal a strategy for planning in a particular domain. One example of this approach is Shavlik's algorithm BAGGER2 (Shavlik, 1990), which uses example solutions and domains and background knowledge in the form of recurrences to learn an algorithm for problem-solving also in the form of recurrences. BAGGER2 was able to learn such algorithms from very few examples, but relied on background knowledge and was unable to identify parallel repetition steps in a transport-domain problem. DISTILL can be seen as an extension of this work but with no

reliance on background knowledge and addressing richly structured parallel repetition.

Our learning of task hierarchies is related to methods for learning macro-operators (e.g., Iba, 1988; Mooney, 1989), in that they explicitly specify the order in which to apply operators, but they do not typically support recursive references. Recent learning work in programming by demonstration such as (Lau, 2001) is also related to our approach. Here, text editor macros are learned by reasoning about how to generalize examples using version space algebras.

Preliminary Results and Conclusions

The POIROT Project is just completing the first of several year-long phases of development. During each phase, our approach is being tested by comparison with the performance of a pool of human subjects given similar background information and practice with a web services interface. For this phase we tested 40 subjects using a demonstration of scheduling four patients that took 41 service calls and 116 parameter assignments. Using automated scoring rules, subjects average about 87% correct in choices of service calls and parameter assignments, but get only about 55% of the right answers. Our preliminary tests with POIROT, combining results from WIT and DISTILL primarily, get very similar results.

These results are preliminary in several important respects. We have not incorporated some of the modules that will induce a stronger hierarchical and goal-directed structure to the learned model (e.g., LIGHT), or detailed preference criteria (DRILL), and so our scoring criteria have not been extended to capture errors due to those factors. We have also not scored the system on a suite of problems that systematically explores the space. More thorough experimental results will be presented at the workshop.

The overall objective of the POIROT project is the development of an architecture that draws on and combines the strengths of a collection of machine learning approaches to solve the difficult problem of learning complex procedural models from a single demonstration. To date, we have been primarily focused on infrastructure development required to support the suite of learning components that will ultimately be part of POIROT. Over the course of the next year, we will be addressing a number of issues relating to the control and coordination of these various learning and reasoning elements, and explicit modeling of the level of certainty of the alternative hypotheses that they produce.

Acknowledgements

We gratefully acknowledge other members of the POIROT team: Paul Cohen, Marie desJardins, Chris Geib, Yolanda Gil, Robert Goldman, Karen Haigh, Pat Langley, Michael Littman, Steve Minton, Tom Mitchell, Karen Myers, Dana Nau, Tim Oates, Jude Shavlik, Katia Sycara, Manuela Veloso and their local staffs for numerous discussions and

contributions to this effort. We also thank the rest of the BBN team including: Michael Atigetchi, Brett Benyo, Karen Haigh, Alice Mulvehill, and John Ostwald for all of their work.

References

- Burstein, M.H. (2004). "Dynamic Invocation of Semantic Web Services that Use Unfamiliar Ontologies," *IEEE Intelligent Systems*, 19(4), p. 67-73.
- Burstein, M., Bussler, C., Finin, T., Huhns, M., Paolucci, M., Sheth, A., Williams, S. and Zarella, M. (2005). "A Semantic Web Services Architecture," *IEEE Internet Computing*, September-October.
- Choi, D., & Langley, P. (2005). "Learning teleoreactive logic programs from problem solving," *Proceedings of the Fifteenth International Conference on Inductive Logic Programming* (pp. 51-68). Bonn, Germany: Springer.
- Cohn, D., Atlas, L. and Ladner, R. (1994). "Improving Generalization with Active Learning," *Machine Learning*, 15(2), p. 201-221.
- Cox, M. T. (1997). "Loose coupling of failure explanation and repair: Using learning goals to sequence learning methods," in D. B. Leake & E. Plaza (Eds.), *Case-Based Reasoning Research and Development: Second International Conference on Case-Based Reasoning* (pp. 425-434). Berlin: Springer-Verlag.
- Cox, M. T., & Ram, A. (1999). "Introspective multistrategy learning: On the construction of learning strategies," *Artificial Intelligence*, 112, 1-55
- Cox, M. T., & Ram, A. (1995). "Interacting learning-goals: Treating learning as a planning task," in J.-P. Haton, M. Keane & M. Manago (Eds.), *Advances in case-based reasoning* (pp. 60-74). Berlin: Springer-Verlag.
- desJardins, M. (1995). "Goal-directed learning: A decision-theoretic model for deciding what next to learn," in A. Ram & D. Leake (eds.), *Goal-Driven Learning*. Cambridge, MA: MIT Press/Bradford Books, pp. 241-249.
- Geib, C. W. and Goldman, R. P. Recognizing plan/goal abandonment. In *Proceedings of IJCAI 2003*, 2003.
- M. Goadrich, L. Oliphant, and J. Shavlik (2004). Learning ensembles of first-order clauses for recall-precision curves. Proc. 14th Intl. Conf. on Inductive Logic Programming (ILP), Porto, Portugal. (Best-paper award winner.)
- Hunter, L. E. (1990b). "Planning to learn," *Proceedings of Twelfth Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Lawrence Erlbaum Associates, pp. 261-276.
- Iba, Glenn, A Heuristic Approach to the Discovery of Macro-Operators. *Machine Learning*, 3(4):285-317, 1989.
- Craig A. Knoblock. Automatically Generating Abstractions for Planning. *Artificial Intelligence*, 68 (2):243-302, 1994.
- R. Korf. Macro-Operators: A Weak Method for Learning. *Artificial Intelligence*, 26 (1):35-77, 1985.
- Lau, T. (2001). *Programming by demonstration: a machine learning approach*. Doctoral dissertation, University of Washington, Seattle.
- Kambhampati, S., & Hendler, J. A. (1992). A validation structure based theory of plan modification and reuse. *Artificial Intelligence*, 55, 193-258.
- David Martin, Mark Burstein, Drew McDermott, Sheila McIlraith, Massimo Paolucci, Katia Sycara, Deborah L. McGuinness, Evren Sirin, Naveen Srinivasan, "Bringing Semantics to Web Services with OWL-S" *WWW Journal* (Forthcoming, 2007)
- Melville, P. and Mooney, R.J. (2004). "Diverse ensembles for active learning," *Proceedings of the Twenty-First International Conference on Machine Learning*, Banff, AB, Canada, p. 584—591.
- Michalski, R. S. (1994). "Inferential theory of learning: Developing foundations for multistrategy learning," in R. S. Michalski & G. Tecuci (Eds.), *Machine learning IV: A multistrategy approach* (pp. 3-61). San Francisco: Morgan Kaufmann.
- Michalski, R. S. & Tecuci, G. (Eds.) (1994). *Machine Learning: A multistrategy approach IV*. San Mateo, CA: Morgan Kaufmann.
- Mooney, Raymond J.: The Effect of Rule Use on the Utility of Explanation-Based Learning. [IJCAI 1989](#): 725-730
- Nau, D., Cao, Y., Lotem, A., & Munoz-Avila, H. (1999). "SHOP: Simple hierarchical ordered planner," *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence* (pp. 968-973). Stockholm: Morgan Kaufmann.
- Paolucci, M, Ankolekar, A., Srinivasan, N., Sycara, K, (2003) "The DAML-S Virtual Machine," *Proceedings of the Second International Semantic Web conference (ISWC 2003)*. Sanibel Island, FL.
- Ram, A., and Leake, D. (Eds.), *Goal-Driven Learning* (pp. 421-437). Cambridge, MA: MIT Press/Bradford Books.
- Salomaki, B., Nejati, N., Choi, D., & Langley, P. (in press). "Learning teleoreactive logic programs from observations," *Proceedings of the 2005 AAAI Fall Symposium on Mixed-Initiative Planning*. AAAI Press.
- Selfridge, O. G. (1959). "Pandemonium: A paradigm for learning," in D. V. Blake and A. M. Uttley (Eds.), *Proceedings of the Symposium on Mechanisation of Thought Processes* (pp. 511-529), London: H. M. Stationary Office.
- Shavlik, Jude W. (1990) "Acquiring recursive and iterative concepts with explanation based learning," *Machine Learning*, 5, p. 39-70.
- Veloso, M. M. (1992). Learning by analogical reasoning in general problem solving. Doctoral dissertation, School of Computer Science, CMU, Pittsburgh, PA.
- F. Yaman and Tim Oates. WIT: Workflow Inference from Traces, AAAI 2007 Workshop on Acquiring Planning Knowledge via Demonstration.