Redesigning a Problem-Solver's Operators to Improve Solution Quality

Eleni Stroulia *
Center for Applied Knowledge Processing
Helmholtzstr. 16
89081 Ulm, Germany
stroulia@faw.uni-ulm.de

Ashok K. Goel
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
goel@cc.gatech.edu

Abstract

The inability of a problem solver to produce solutions of a desired quality often may lie in the incorrect design of its operators. In this paper, we describe a method which, given a problem solver that produces a poor solution for a given problem, and given the desired solution for it, uses a model of the problem-solver's processing and knowledge to identify faults in the specification of its operators and to appropriately redesign them. This method is based on the structure-behavior-function (SBF) model of problem solving that explicitly captures the functional semantics of the problem-solver's tasks, the compositional semantics of its problem-solving methods that combine the operators' inferences into the outputs of the overall task, its domain knowledge, and the "causal" interdependencies between its tasks, methods and domain knowledge. We illustrate and evaluate this learning method through AUTOGNOSTIC.

1 Introduction

Research on unification of problem solving and learning has led to the development of several learning methods for improving problem-solving efficiency (e.g., [Mitchell *et al.*, 1981]). These methods typically assume that the set of available problem-solving operators is both complete and correct. The methods instead improve the control of processing, either by compiling sequences of operators into macro-operators or by acquiring better heuristics for their selection. In this paper, we are interested in the complementary learning goal of improving the quality of the solutions a problem solver generates. Modifying the control of processing is, in general, insufficient for attaining this goal because the reason why a problem solver produces poor solutions often is that its operators are incorrect. The specific research issue is as follows: given a problem solver that fails to produce solutions of a desired quality, and given desired solutions for problems for which it produced poor-quality solutions, what combination of knowledge and processing enables the identification of the incorrectly specified operators and their redesign, so that problem solver can deliver solutions of the desired quality?

To address this problem, we have developed a learning method based on a structure–behavior–function (SBF) model of a problem-solver's processing and knowledge. This model captures (i) the functional semantics of the problem-solver's tasks, (ii) the compositional semantics of its methods which recursively synthesize the inferences drawn by its operators into the outputs of its overall task, (iii) the domain knowledge available to it, and (iv) the "causal" interdependencies between its tasks, methods and domain knowledge. This model-based learning method is implemented in AUTOGNOSTIC, a "shell" which provides a language for representing SBF models of problem solvers, and mechanisms for monitoring the problem solving, receiving feedback on the result, and, in case of failure, assigning blame and repairing the problem solver. AUTOGNOSTIC's method for assigning blame for specific kinds of failure and identifying what operators to modify is described in [Stroulia and Goel, 1996]. In this paper, we focus on its method for redesigning the incorrect operators. AUTOGNOSTIC's integration with ROUTER, a path-planning system, is used to illustrate the method.

2 SBF Models of Problem Solving

SBF models analyze the problem-solver's task structure its domain knowledge and their interdependencies. In this framework, the problem-solver's tasks constitute the building blocks of its problem-solving mechanism. The methods that it employs decompose its complex overall tasks into simpler subtasks. These, in turn, get recursively decomposed into even simpler subtasks until they become elementary reasoning steps, i.e., *"leaf" tasks*, directly accomplished by the problem-solver's domain operators.

A task is specified as a transformation from an input to an output information state. It is characterized by the type(s) of information it consumes as input and produces as output, and the nature of the transformation it performs between the two. A task's *functional semantics* partially defines the task's intended correct behavior, by specifying the nature of the task's information transformation; it is expressed in terms of do-

*In the Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence Nagoya, Japan, August 23-29, IJCAI'97.

main relations among the task's inputs and outputs. [1] Notice, that functional semantics is not simply a set of pre- and post-conditions for the application of an operator; it is rather a functional concept specifying how the task's output relates to its input. For a non-leaf task, the functional semantics of the subtasks into which the task is recursively decomposed, and the ordering relations that the decomposing methods impose over them, constitute a partial description of a correct reasoning *strategy* for this task.

Methods can be thought of as general plans for how the solutions of low-level subtasks get combined to deliver those of higher-level tasks. Each method captures the *semantics of the composition* of a set of lower-level subtasks into a higher-level task in terms of control, and information interdependencies. The tasks' control interdependencies are a set of relations partially ordering their executions; their information interdependencies are a set of information producer-consumer relations among them.

Finally, the SBF model of a problem solver captures the problem-solver's domain ontology, in terms of the types of objects that the problem solver knows about, and the relations applicable to them.

The Case Study: ROUTER, [Goel *et al.*, 1994], the case-study problem solver which will be used in this paper to illustrate AUTOGNOSTIC's model-based learning method, is a path planner. Its spatial world model is organized in a neighborhood-subneighborhood hierarchy. High-level neighborhoods describe large spaces in terms of major streets and their intersections, and get refined into lower-level neighborhoods which describe both major and minor streets and their intersections but over smaller spaces. In addition to its world model, ROUTER contains a memory of past path-planning cases, also organized around the neighborhood-subneighborhood hierarchy. Figure 1 diagrammatically depicts part of ROUTER's SBF model. It illustrates the SBF specification of part of its task structure and domain knowledge.

ROUTER's task, *path-planning*, is to find a path from an initial to a goal location. To that end, ROUTER first identifies the neighborhoods in which these locations belong (*elaboration* subtask). Then, it searches in its path memory, for a path that is close to the current problem (*retrieval*). If the two given locations belong in the same neighborhood, ROUTER may use the *intrazonal-search* method to *search* for a path between them. This method is essentially a breadth-first search within the common neighborhood of the two locations. Initially, ROUTER sets up as its current location the initial location, and initializes its *tmp-path* to contain only this location. Then, by repeating the *path-increase* subtask, it incrementally expands the *tmp-path*. If, at some point, the *tmp-path* reaches the goal location, ROUTER assigns its value to the *path* and returns it as the solution. If the given locations do not belong in a common neighborhood, ROUTER can either perform a hierarchical search in its neighborhood organization (*interzonal-search* method) or it can use the path it retrieved from its memory, as the basis for solving the current problem (*case-based* method).

[1] The SBF language presently allows up to ternary relations and a single qualification level for the representation of functional semantics.

3 Redesigning Incorrect Operators

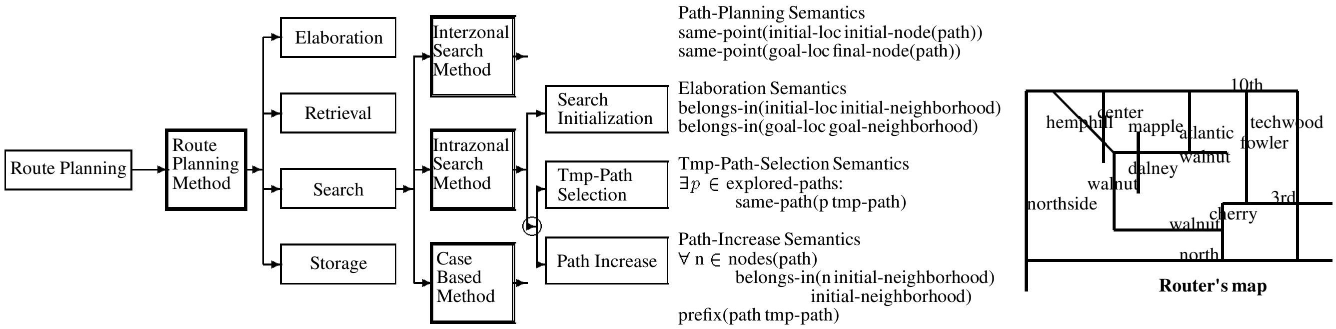
The inability of a problem solver to produce solutions of the desired quality is revealed when, for some particular problem, it receives as feedback a desired solution different from the one it actually produced. The first subtask of our model-based learning method is to assign blame for the failure to a specific operator so that it can then proceed to redesign it.

3.1 Assigning Blame

Blame assignment is performed through a successively focused examination of the tasks and methods involved in the production of the solution. Beginning with the problem-solver's overall task, the blame-assignment method evaluates whether the feedback and the task's actual input validate the task's functional semantics. If this is the case, it infers that the task in question should have produced the feedback, and the reason why it did not must lie within the subtasks in which it was decomposed. Based on the problem-solving trace, the last such subtask is identified and the blame assignment focuses on it. If the functional semantics of a particular task are violated by the feedback and its actual input, the blame-assignment method attempts to infer alternative inputs which would make the feedback value valid with respect to the task's semantics. If such alternative inputs can be found, the actual inputs are considered undesirable, and the blame-assignment focuses on the earlier subtask responsible for their production.

Identifying Over-Constrained Operators: If the blame assignment reaches a leaf task whose semantics is violated by the feedback solution and its actual input, and no alternative input can be found to make the feedback valid, then it postulates as one potential cause for the problem-solver's failure the *over-constrained semantics* of the task. The feedback solution exemplifies a set of quality requirements which, although in accordance with the overall behavior expected by the problem solver, conflict with the specification of the behavior delivered by a low-level operator. Thus, the feedback reveals an error in the information transformation that this particular operator was designed to accomplish in the context of the overall task, since it prevents the production of an acceptable and desired solution.

Identifying Under-Specified Operators: Alternatively, the blame-assignment may reach a leaf task whose semantics is validated by the output desired of it as well as its actual output. This situation implies that the problem-solver's task structure is not sufficiently specified to produce the right quality of solutions. Thus, the blame assignment postulates that the cause of the failure might be the *under-specified semantics*



TYPES OF DOMAIN OBJECTS

	PATH		INTERSECTION
id_test	path-equivalence		intersection-equivalence
domain	–		intersection-domain
attrs	nodes	length	streets
	self	length(self)	self
	(list-of intersection)	number	(list-of 2 street)
rels	begins-in-neighborhood		connected-to
	ends-in-neighborhood		belongs-in

TYPES OF DOMAIN RELATIONS

	PREFIX	CONNECTED-TO	BELONGS-IN
input_args	path	intersection	intersection
output_args	(list-of path)	(list-of intersection)	neighborhood
truth_table	–	–	neighborhoods-of-ints-table
predicate	is-prefix	adjacent-ints	–
inv_predicate	prefix-of	adjacent-ints	–

Figure 1: Fragment of ROUTER's SBF model: part of its task structure and domain knowledge.

of the task, which allows both the actual and the desired values to be produced, when only the latter conforms with the requirements on the quality of the problem-solver's solutions, as exemplified by the feedback.

3.2 Redesigning an Incorrect Operator

To address the learning tasks that arise from identifying an operator, *under-specified* or *over-constrained* with respect to the overall problem-solving task in service of which it is employed, a learning method must be able to formulate new functional semantics for this operator. That is, the learning method must be able to discover new functional concepts to characterize the relation of the operator's output to its input. In the former case, the new semantics will refine(specialize) the functionality of the faulty operator, in the latter one, it will extend(generalize) it. To that end, the model-based learning method we present in this paper relies first, on the examples of the behavior desired of the operator in the context of the overall task as exemplified by the feedback, and second, on its comprehension of the domain knowledge generally available to the problem solver.

Let us now see how the examples of the operator's desired behavior are generated. As it successively refines the focus of the investigation from the overall task to its increasingly specific subtasks, the blame assignment also "translates" the behavior desired of this overall task into behaviors desired of its subtasks. This translation occurs in two steps. In the first, when the investigation is focused from one task to the last subtask set up by the method used to accomplish it, the feedback becomes the output desired of the more specific subtask. In the second, when the investigation moves from a subtask to an earlier one, the feedback enables the inference of alternative inputs for the current task which, then, become the outputs desired of the earlier one. Thus, by the time the blame assignment reaches an incorrect (i.e., under-specified or over-constrained) operator, it has also produced an example of the behavior desired of this operator in the actual problem-solving session, namely an input-output tuple, which, if produced by the operator, could lead to the production of the feedback desired of the overall task. In the case of an under-specified operator, the *actual-input desired-output* tuple constitutes an example of the behavior desired of it, where the *actual-input actual-output* tuple constitutes an example of undesired behavior. In the case of an over-constrained operator, both the *actual-input desired-output* and the *actual-input actual-output* tuples constitute examples of desired behavior. Having such a set of positive and negative examples of the information transformation that the operator should perform, the next goal of the learning method becomes to discover a semantics specification to characterize them in an abstract way.

Here, let us discuss how the comprehension of the domain knowledge is used in the redesign of faulty operators. The SBF specification of the problem-solver's tasks' semantics is based on the domain relations known to the problem solver. In a sense, these domain relations provide the "terms" in which the information-transformation functions of the problem-solver tasks are expressed. By specifying the domain relations available to the problem solver, independently of whether or not they are used by its current task structure, the SBF model specifies the range of inferences that are possible in the problem-solver's domain. This specification of potential inferences enables our model-based learning method to recognize new uses for the problem-solver's knowledge in characterizing the alternative information-transformation functions desired of its currently incorrectly specified operators.

Figure 2 depicts the algorithm for discovering a new semantics with which to characterize a set of positive and negative examples of an operator's desired behavior. The learning method consists of two basic steps. First, it establishes a hypothesis space in which to search for possible alternative

```
New-Semantics (info, pos-cases, neg-cases, task, top-task)
domain-rels := the domain relations referring to the wo in which info belongs          (1)
avail-info := the types of information produced before the task in the most specific
method in which task belongs.                                                           (2)
FORALL rel in domain-rels
      wo2 := the world-object type that rel refers to, in addition to wo(info)
      FORALL i2 in avail-info that wo(i2) = wo2
            instantiate rel with info and i2                                            (3)
            IF FORALL c in pos-cases true(rel(v(i,c) v(i2,c)))
                AND FORALL c in neg-cases false(rel(v(i,c) v(i2,c)))
            THEN include rel in out-rels                                                (4)
IF there are no out-rels
THEN FORALL attr in attr(i)
      out-rels := New-Semantics (attr, pos-cases, neg-cases, task, top-task)            (5)
IF there are no out-rels
THEN avail-info := the types of information produced before the task in the next most
specific method in which task belongs.
       GOTO line 2                                                                      (6)
```

Figure 2: The Algorithm for Discovering new functional semantics for incorrectly defined problem-solving operators

semantics for the incorrectly specified operator, and second, it successively evaluates the hypotheses in this space and collects these semantics that correctly characterize the behavior desired of the operator.

Establishing a Hypothesis Space: Given the representation of the functional semantics of a task in the SBF language, the formulation of the semantics hypothesis space requires first, the identification of the possible domain relations on which this semantics may be based, and second, the identification of the information types which, in addition to the operator's output, may be mentioned in the semantics.

To address the first requirement, the method uses the SBF specification of the operator's output information, to infer in which type of object it belongs. Next, based on the SBF specification of this object type, it identifies the set of domain relations relating this object type with other object types whose instances can be found among the set of available information types (line 1 of the algorithm shown in Figure 2). To address the second requirement, the method identifies the set of information types, available to the problem solver when the incorrect operator is applied. To that end, it identifies the most specific method which invokes the incorrect operator, and then these subtasks of the method which should have already been accomplished before the operator is invoked. Then, the types of information that can be used in the description of the new functional semantics are the types of information consumed and produced by these subtasks (line 2). The final step involves the formulation of hypotheses for the new semantics, using the set of available information types and the set of domain relations that may apply to them. For each domain relation, a semantics is formulated using as arguments the operator's output information and every other available information whose type conforms with the type of the relation's other argument (line 3).

Evaluating the Semantics Hypotheses: Having formulated a set of potential semantics for the incorrect operator, the learning method proceeds to evaluate which of them actually characterize the operator's desired behavior, by evaluating which ones are validated by the positive examples of the operator's desired behavior and fail for the negative examples (line 4). Any of the semantics relations that fulfill these requirements can be used to respecify the incorrect operator. If no such semantics relation is found, the learning method uses the SBF specification of the object type of the operator's output to infer its attributes and their corresponding values for the output actually produced and the output desired of it, and proceeds to repeat the above step sequence, in order to identify semantics in terms of the output-information's attributes (line 5). If again no semantics is found, the learning method extends the set of available information types by collecting the information types available in the context of a more general method subsuming the incorrect operator (line 6).

4 An Example in Router

Let us now illustrate the above method with an example from Router. Router is given the problem of going from *(10th center)* to *(walnut dalney)*. It produces the path *(center 10th) (10th atlantic) (atlantic walnut) (walnut dalney)*, and is given the shorter path *(center 10th) (center mapple) (mapple dalney) (dalney walnut)* as feedback.

Assigning Blame: Autognostic first evaluates whether or not the feedback path is valid with respect to the functional semantics of Router's overall *path-planning* task (as shown in Figure 1). The desired path begins at the initial intersection and ends at the final one, therefore, Autognostic infers that the desired value is within the class of solutions that Router was intended to produce in this particular problem. Thus, it successively refines the focus of its investigation to increasingly specific subtasks of the *path-planning* task which produce the *path*, that is, *search* and subsequently *path-increase*.

As it evaluates the semantics of *path-increase*, Autognostic notices that the feedback path does not have as its subpath the task's input *tmp-path*. The *tmp-path* selected in the last repetition of the loop was *(center 10th) (10th atlantic) (atlantic walnut)*, and it is not a prefix of the desired path. At this point, Autognostic infers that the desired path could not possibly have been produced by the *path-increase* task given the information it received as input. To infer alternative values for the input *tmp-path* which could enable the task at hand to produce the desired path, Autognostic uses the predicate for the inverse *prefix-path* relation, and infers that *(center 10th) (center mapple) (mapple dalney)* is one prefix of the desired path, which, if given as *tmp-path* to the *path-increase* subtask, could result in the feedback's production. Thus, the cause of the failure must lie within the subtask that produced the "wrong" *tmp-path*, i.e., the task *tmp-path selection*.

Autognostic focuses its investigation towards identifying why this earlier subtask did not produce the desired *tmp-path*. It evaluates its functional semantics for the *tmp-path* desired of it, and notices that it is satisfied. Indeed, the desired *tmp-path* belongs in the set of *possible-paths*, and therefore, it could have been produced by *tmp-path selection*. Since this is a leaf task, accomplished by an operator, AUTOGNOSTIC infers that the specification of the *tmp-path selection* operator is too general with respect to the quality of solutions desired of the overall *path-planning* task.

Redefining the *tmp-path selection* **Operator:** Indeed, if "short length" is a quality desired of the paths that ROUTER produces, then its intrazonal-search method, which is essentially a breadth-first search within a neighborhood, is inappropriate. Instead, a depth-first search method should be used to actively prefer short paths, and in order for that to happen, the *tmp-path selection* should be redesigned to prefer the short paths in the *possible-paths* list. Let us now see how AUTOGNOSTIC reaches this conclusion.

The blame-assignment step has localized the under-specified *tmp-path selection* operator. It actually produced the *tmp-path (center 10th) (10th atlantic) (atlantic walnut)*, when it should have produced *(center 10th) (center mapple) (mapple dalney)* in order for the overall *path-planning* to produce the feedback.

The next step of the learning method is to collect the types of information available to ROUTER when the *tmp-path selection* is accomplished. This set includes the inputs and outputs of the *search-initialization* subtask, the only subtask of the *intrazonal-search* method (i.e., the method invoking the incorrect operator) which precedes the operator. Thus, the set consists of *initial-zone, initial-int, possible-paths*. These are the information types that can be used as terms in the new semantics of the *tmp-path selection* operator, and they are a *neighborhood*, an *intersection*, and a *path* respectively. Next, AUTOGNOSTIC collects the domain relations relating the above types of objects with *path*s, and formulates a set of potential semantics relations with these domain relations relating the *initial-zone*, or *initial-int* or *possible-paths* with *tmp-path*. Finally, it proceeds to evaluate whether any of these semantics characterizes the behavior desired of *tmp-path selection*. None of them does, so AUTOGNOSTIC proceeds to search for semantics based on the attributes of the available information types. In this effort, it identifies a single candidate for the role of *tmp-path selection* semantics, namely the relation *forall p in possible-paths, length(tmp-path)* $\leq$ *length(p)*. AUTOGNOSTIC proceeds to redefine the *tmp-path selection* operator to perform the function suggested by its new semantics, i.e., to select from the set of *possible-paths* the shortest one to expand [2]. Then, after presenting ROUTER with the same problem for a second time and producing the desired path as output, AUTOGNOSTIC verifies the appropriateness of the new semantics.

5 Experimentation and Evaluation

Two important issues for unified problem-solvers and learners, such as AUTOGNOSTIC, are how well do their learning methods accomplish their learning tasks in individual problem-solving episodes, and whether the learning converges to a stable design for the problem-solver. To test the efficacy of AUTOGNOSTIC's learning method in individual episodes, we gave AUTOGNOSTIC–on–ROUTER three (3) sequences of forty (40) randomly-generated path-planning problems. For each problem, we also gave as feedback the shortest path, where path length was measured by the number of discrete path segments in a path plan. We found that for all failed problem-solving episodes, in which ROUTER generated a path longer than the desired one, AUTOGNOSTIC succeeded in assigning blame and appropriately repairing ROUTER. However, since AUTOGNOSTICis an incremental learner, it suffers from the familiar problem of over-generalization (and over-specialization) of the functional semantics on the basis of a single example.

To test the long-term effectiveness of the learning process, after each of the above training sets, we gave AUTOGNOSTIC–on–ROUTER a sequence of one hundred and fifty (150) randomly generated path-planning problems. For each training set, and for each problem in the test set, we compared the quality of the solutions produced by the original and the modified ROUTER. The table below summarizes the results:

Table 1: Results on Quality-of-Solution Improvement.

training set	problems with better path quality	problems with worse path quality	sign test	paired t-test
1	55	13	$1.3\ 10^{-7}$	0.0002 (t=3.88)
2	53	14	$8.8\ 10^{-7}$	0.0000 (t=5.17)
3	44	30	0.06	0.14 (t=1.49)

In these experiments, AUTOGNOSTIC modified ROUTER's *tmp-path selection* operator, as we described above, and two additional operators invoked by the *case-based* and the *interzonal-search* methods. This enabled ROUTER to perform a greedy search for short paths, which, for an almost canonical street network, returns the shortest path in most cases. In addition, AUTOGNOSTIC extended the size of ROUTER's neighborhoods to enable it to search through larger spaces in order to find shorter paths. That AUTOGNOSTIC modified the same three operators for all the different problems in the three experiments suggests that AUTOGNOSTICis able to determine that only some operators are "faulty" in ROUTER's design. It also indicates that AUTOGNOSTIC does not keep on modifying the problem solver indefinitely, but stops when, after some exploration and redesign, a good enough design is found.

6 Related Research

Although much of past research on unification of problem solving and learning has focused learning methods for improving problem-solving efficiency, recent work in the Prodigy [Carbonell *et al.*, 1989] framework has also explored learning of problem-solving operators. Perez [1994] describes a learning method for improving the quality of a problem-solver's solutions. This method uses both a trace of the problem solving that led to the solution of less than the desired quality and a trace of hypothetical problem solving that would lead to a solution of the desired quality. It assumes that the problem-solver's operators are correct and the

[2] The "reprogramming" of the operator is done manually.

cause of its failure to produce good solutions lies in its incorrect operator-selection heuristics. Gil [1994] describes a method for identifying missing pre- or post-conditions in a problem-solver's operator specification. This is similar to the learning task of AUTOGNOSTIC, but different, since it does not modify the information transformation the incorrect operators perform. Wang [1995] describes a method for acquiring operators from problem-solving traces, where the GPS-like operators again are specified in terms of pre- and post-conditions, without any functional specification of the input-output relations.

While the learning methods in the Prodigy framework are trace-based, the learning methods in our theory are model-based. In particular, AUTOGNOSTIC's learning methods use SBF models of problem solvers. We initially developed SBF models to capture the functional, compositional and causal semantics of physical devices [Goel 1991]. In AUTOGNOSTICwe have used the SBF modeling methodology to explicitly specify the tasks, methods, control of processing, and domain knowledge of problems solvers in a manner consistent with Chandrasekaran's framework of Generic Tasks and Task Structures [Chandrasekaran 1989]. CommonKADS [Wielinga *et al.* 1992] is another task-oriented framework for analyzing and describing problem solving. A major advantage of analyzing problem-solvers in terms of tasks and methods is that enable specification of the organization of problem solving, which in turns localizes the learning. Alternative model-based methods, such as that of Castle [Freed *et al.*, 1992], can only describe problem solvers performing a linear sequence of tasks, and, in addition, are limited in the learning tasks they address.

7 Conclusions

We have described a learning method based on SBF models of problem solvers which is able to identify incorrectly specified problem-solving operators, and to appropriately redesign them. This method relies on the specification of the functional semantics of the problem-solver's tasks, the compositional semantics of the methods that combine the inferences of its low-level operators into the outputs of its high-level tasks, and its domain knowledge.

The learning method uses the specification of the functional semantics of the problem-solver's tasks to establish the range of behaviors that the problem-solver's design is intended to deliver. Also, by evaluating the functional semantics of the problem-solver's subtasks relative to the solution desired of the problem solver, the learning method identifies which of the operators are incorrect with respect to the desired solution. This also enables the learning method to infer the behaviors desired of these subtasks in specific problems, which are then used as the positive examples in formulating their new functional semantics. Finally, the representation scheme for functional semantics in the SBF language enables the learning method to define the dimensions of the space of the new functions.

The SBF specification of the compositional semantics of the problem-solver's methods enables the learning method to limit the number of information interdependencies it examines, specifically by investigating only the tasks involved in the production of the undesired output. In addition, it enables the incremental specification of the information types in terms of which the new functional semantics of the incorrect operator can be specified. Finally, the specification of the problem-solver's domain ontology enables the learning method to establish the set of information relations that can be used in the specification of these new functions.

Acknowledgments

This paper has benefited from numerous discussions with members of the Intelligence and Design research group at Georgia Tech. This work has been supported in part by research grants from NSF (IRI-92-10925 and DMI-94-20405) and ONR (research contract N00014-92-J-1234).

References

[Carbonell *et al.*, 1989] Carbonell, J.G., Knoblock, C.A. and Minton, S. Prodigy: An Integrated Architecture for Planning and Learning. In *Architectures for Intelligence*, K. VanLehn (ed.), Hillsdale, NJ: Erlbaum.

[Chandrasekaran 1989] Chandrasekaran, B. Task Structures, Knowledge Acquisition and Machine Learning. *Machine Learning* 4:341-347, 1989.

[Freed *et al.*, 1992] Freed, M., Krulwich, B., Birnbaum, L. and Collins, G. Reasoning about performance intentions. In *Proceedings of the Fourteenth Conference of Cognitive Science Society*, pages 7-12, Hillsdale, NJ: Erlbaum.

[Gil, 1994] Gil, Y. Learning by Experimentation: Incremental Refinement of Incomplete Planning Domains. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 87-95. Los Altos, CA: Morgan Kaufmann.

[Goel 1991] Goel, A. Model Revision: A Theory of Incremental Model Learning. In *Proceedings of the Eighth International Conference on Machine Learning*, pages 605-609, Chicago, June 1991. Los Altos, CA: Morgan Kaufmann.

[Goel *et al.*, 1994] Goel, A., Ali, K., Donnellan, M., Gomez, A. and Callantine, T. Multistrategy Adaptive Navigational Path Planning. *IEEE Expert*, 9(6):57-65, 1994.

[Mitchell *et al.*, 1981] Mitchell, T.M., Utgoff, P.E., Nudel, B. and Banerji, R.B. Learning problem-solving heuristics through practice. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pages 127-134.

[Perez, 1994] Perez, M.A. The goal is to produce better plans. In *Working Notes of AAAI Spring Symposium Goal-Driven Learning*, March 21-23, 1994, Stanford University; AAAI Press.

[Stroulia and Goel, 1996] Stroulia, E. and Goel, A. A Model-Based Approach to Blame Assignment: Revising the Reasoning Steps of Problem Solvers. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 959-965, Portland, Oregon, August 4-8. AAAI Press.

[Wang, 1995] Wang, X. Learning by Observation and Practice: An Incremental Approach for Planning Operator Acquisition. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 549-557. Los Altos, CA: Morgan Kaufmann.

[Wielinga *et al.* 1992] Wielinga, B.J., Schreiber, A.Th., and Breuker, J.A. KADS: A modelling approach to Knowledge Engineering. *Knowledge Acquisition*, 4(1):5-53.