Generalization for Explanation-based Schema Acquisition

Paul O'Rorke

Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

ABSTRACT

This paper is about explanation-based learning for heuristic problem solvers which "build" solutions using schemata (frames like scripts) as both "bricks" and "mortar". The heart of the paper is a description of a generalization method which is designed to extract as much information as possible from examples of successful problem solving behavior. A related generalizer, (less powerful but more efficient), has been implemented as part of an experimental apprentice.*

I INTRODUCTION

Many knowledge-based AI systems have used schemata (knowledge packets such as frames or scripts) as the basis of computational models of understanding [1], planning [2] or other problem solving [3], but very few of these systems have been capable of generating their own schemata. As a result, most schema-based systems have been unable to automatically profit from their experiences, so that the main way of improving their performance has been by laboriously hand-coding new schemata.

At the University of Illinois, a small group led by Prof. Gerald DeJong has been exploring and automating a solution to this knowledge-acquisition bottleneck: a particular brand of explanation-based learning called "explanatory schema acquisition (ESA)." [4,5]

This paper describes the explanation and generalization methods underlying explanatory schema acquisition in the context of our first complete implementation of an experimental apprentice. The apprentice, (named MA), contains a heuristic search schema-based problem solver specializing in interactive human-oriented theorem proving. Like any other apprentice, MA starts life with very limited problem solving ability. Initially, MA can only make tiny contributions to most problem solving efforts; its master must supply the insights which lead to successful proofs. At first, MA "merely" observes the master's behavior, but MA recognizes when this behavior leads to success. Then by generalization based on analysis of the reasons for success MA learns new schemata and heuristics for their use.

II PROBLEM SOLVING WITH SCHEMATA

Schema-based problem solvers are goal directed systems which aim to construct schemata satisfying given constraints. Of course, the details of schemata will vary from one domain to another but in general schemata used in explanation-based schema acquisition systems are comprised of parameters (variables), constraints on parameters (which may function as "slots"), and dependency relations between the constraints.

A schema-based problem solver makes progress toward its goal by instantiating general schemata called prototypes. Instantiation is accomplished by invoking a prototype (copying it with new, unique names for parameters) and binding parameters. Parameters may only be bound to or identified with objects subject to the constraints. Two kinds of prototype are used to build solutions: primitive schemata and schematic forms. A schematic form is a schema which is used to combine existing schemata into a new composite schema; it has parameters which are constrained to be filled by other schemata.

MA has a primitive proof prototype which plays the role of the assumption axiom schema of Manna's Gentzen style natural deduction system [6]. Given a set of hypotheses, the assumption axiom can be used to infer a desired conclusion when it is a member of the given set. The parameters associated with AssumptionAxiom schemata are Self, A and Gamma. A is constrained to be a WFF (well-formed formula) and Gamma must be a SET-OF-WFFS. The Self parameter has the constraint (PROOF Self OF A FROM (Union Gamma {A})) which depends on the other constraints. The constraints associated with an instance of a prototype are represented as assertions in a database and the dependencies between the constraints are represented as data-dependencies as described and illustrated in [7]. The dependency graph associated with an instance of assumption axiom represents the fact that the instance (denoted by "Self") is a complete proof (of A from the set of WFFs derived by adding A to Gamma) if and only if A is a WFF and Gamma is a set of WFFs.

* This report describes work done in the AI group of the Coordinated Science Laboratory at the University of Illinois at Urbana-Champaign. The work was supported by the National Science Foundation under grant NSF IST 81-20254.

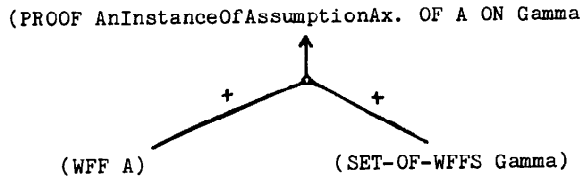

Fig. 1: AssumptionAxiom Constraints & Dependencies

Not all MA's schemata are primitive like assumption axiom. Complex schemata can be constructed using "inferential forms." For example, MA has schematic forms (corresponding to the "Or Introduction" rules of [6]) which facilitate the construction of proofs of disjunctions. If we have a proof of A we can plug it into an instance of OrIntroductionA to get a proof of A OR B for any WFF B. A similar OrIntroductionB schema can be used to construct a proof of a disjunction out of a proof of the second disjunct.

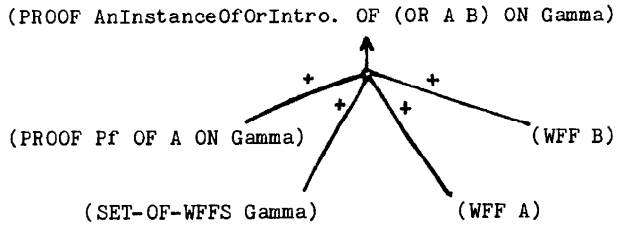

Fig. 2: OrIntroduction Constraints & Dependencies

MA also has an inferential form corresponding to the Elimination of Assumption rule. If we have two proofs of A, one based on some assumptions plus B and another on the same assumptions plus NOT B then we can plug the proofs into this form to get a proof of A that doesn't depend on B.

Of course, it's not enough simply to have a collection of passive schemata: one must also know how to use them! MA is a heuristic schema based problem-solver because heuristics control the instantiation of schemata. The heuristics have conditions which only allow invocation of a schema when it is sure to help achieve the goal. For example, MA has a heuristic which causes it to instantiate AssumptionAxiom to achieve goals of the form (PROOF GoalPF OF A FROM (Union Gamma {A})) because AssumptionAxiom is sure to achieve such goals. OrIntroduction schemata are invoked for goals of the form (PROOF GoalPF OF (OR A B) FROM Gamma) because they reduce such goals to simpler goals of proving one or the other disjunct. On the other hand, just because a schema can be applied in a given situation is no reason that it should be, so MA does not instantiate EliminationOfAssumption just because it sees a goal of the form (PROOF GoalPF OF A FROM Gamma).

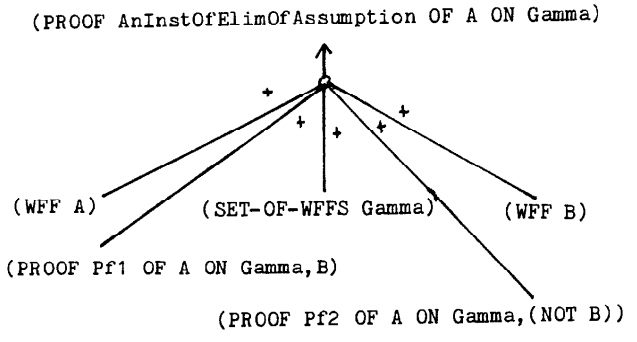

Fig. 3: EliminationOfAssumption Constraints

Narrowing the conditions under which schemata are invoked has the advantage of minimizing search, but we pay a price in generality (in this case our theorem prover is rendered incomplete). In other words, when the problem solver can solve a problem it will do so efficiently but there will be soluble problems which it can not solve. Initially MA, (like many people), will not know what to do if you ask it to prove P OR NOT P is a tautology (i.e. to construct a schema named GoalPF under the constraint that (PROOF GoalPF OF (OR P (NOT P)) FROM Empty-Set)). None of MA's heuristics are applicable to this goal, so the apprentice program gives up and waits for its master to give it a hint.

To set the stage for the next section, assume that the user sees how to build the desired proof. Assuming the user applies EliminationOfAssumption toward achieving the goal, MA will fill in the needed OrIntroduction(A and B) and AssumptionAxioms hooking them up as illustrated in figure 4. More importantly, the achievement of the goals involved will trigger the generalization algorithm, which will make sure MA is not at a loss when confronted with similar problems and subproblems in the future!

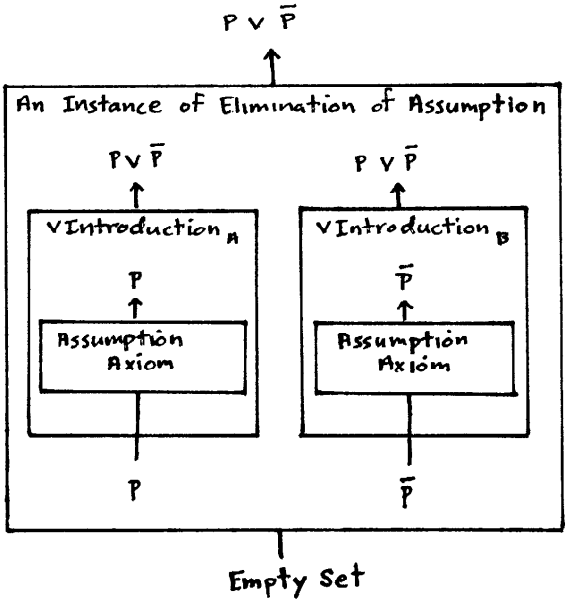

Figure 4: A proof that P OR (NOT P) is a tautology

III GENERALIZATION

The goal of ESA learning is to improve the performance of schema-based problem solvers. One way of maximizing this improvement is by extracting as much knowledge as possible from given examples of success. When knowledge is encoded in schemata and heuristics representing associations between different goals and methods of achieving them, ESA can enable the problem solver to achieve as many new goals as possible by maximizing

1) the number and
2) generality of heuristics and schemata extracted from each example.

Extracting many schemata can enable the problem solver to achieve subgoals which were achieved during the construction of complex examples. In the P OR NOT P example, we should not only have a heuristically invoked schema generalized from the proof of figure 4, but we should also extract general schemata from subproofs such as the proof of P OR NOT P from P. In addition it is sometimes desireable to extract new schematic forms from example schemata. This amounts to learning new ways of combining schemata to achieve goals. This makes it possible for MA to learn new "inferential forms" (ways of forming new proofs from existing proofs) like the derived rules of inference in [6]. Please see the expanded version of this report for a discussion of these issues [8].

This section focuses on maximizing the quality (generality) rather than the quantity of new heuristics and schemata. Maximizing generality enables the problem solver to achieve many goals which differ in insignificant ways from the goals successfully achieved in an example. ESA maximizes generality by minimizing constraints associated with examples: dropping all irrelevant details due to idiosyncracies of the example while retaining important facts.

For example, consider the composite schema used in the proof of P OR NOT P from the Empty-Set. It is obvious that the same composition of instances of EliminationOfAssumption, OrIntroduction(A and B), and AssumptionAxiom could be used to prove conclusions other than P OR NOT P from sets of hypotheses other than the Empty-Set. Most people realize this composition could be used to prove any conclusion of the form A OR NOT A (A need not be the particular WFF: P) from an arbitrary (possibly non-empty) set of hypotheses.

ESA uses explanations of the reasons for success as the basis for generalization and for determining the condition under which the generalized schemata should be invoked. The explanations are embodied in dependency networks generated during the process of solving a problem. Figure 5 shows part of the dependency network underlying the P OR NOT P example.

An ESA algorithm computes the condition which determines when a novel schema should be invoked by examining this sort of explanation and collecting facts crucial to the success of the schema. The following taxonomy is used to seperate the important "wheat" from the irrelevant "chaff."

A TAXONOMY OF CONSTRAINTS

Essential constraints form integral parts of explanations of success and must be incorporated into the results of generalization. There are two types of essential constraint:

> Essential Inter-schema constraints connect component schemata together into a complex schema. Technically, these are the assertions which support the goal achievement by identifying parameters of instances of schematic forms with instances of prototypes.
>
> Essential Intra-schema constraints ensure that each component of a complex schema is "complete." Technically, these are the immediate supporters of the "self" constraints which support the achievement of the goal.

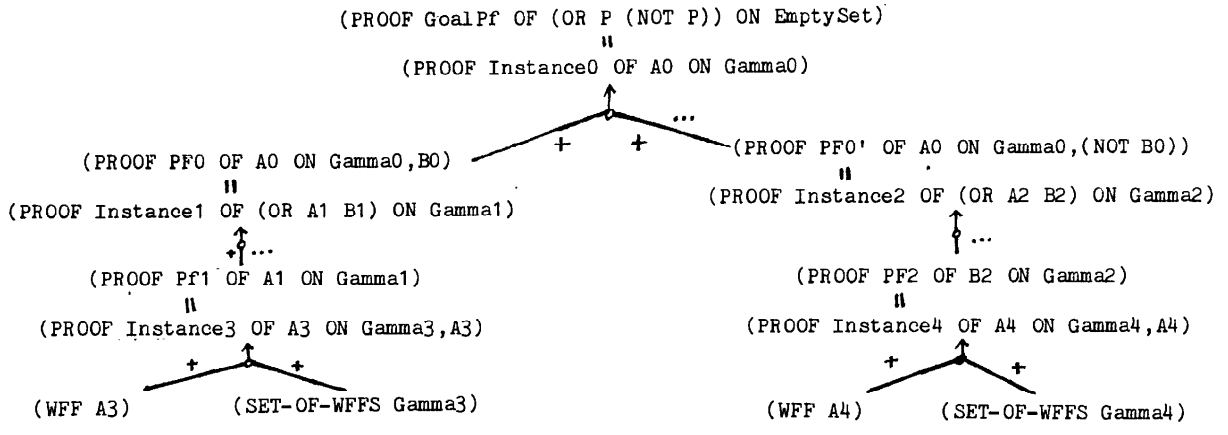

Fig. 5: Dependency Net Underlying P OR NOT P Example

Optional constraints are "forced" or implied by essential constraints. They need not be included but should be. They don't alter generality but improve efficiency. Technically, any assertion which has some justification depending purely on essential constraints is in this class.

Extraneous constraints include most instantiation bindings and all implications based at least in part on extraneous constraints. Technically these are just defined to be the non-essential non-optional constraints.

In the P OR NOT P example, extraneous constraints include the identification of the conclusion as P OR NOT P and the identification of the set of hypotheses as the Empty-Set. In fact, one may use an arbitrary set of hypotheses and the conclusion does not even have to be a disjunction of the form A OR NOT A. "Anything goes" so long as the essential constraints (which hold the composite schema together and which ensure that each component is legally instantiated) are not violated.

It turns out that ambiguous constraints allow the ESA generalization method sketched in this paper to learn more from the P OR NOT P example than most people [8]. This is because most people don't realize that the same composite schema applied to proving P OR NOT P from Empty-Set can also be used to construct proofs:

OF (A OR B) FROM (Union Gamma {A})
OF (A OR B) FROM (Union Gamma {B})
OF (A OR B) FROM (Union Gamma {A} {B})

Unfortunately, our first implementation shares this fault: it only learns to invoke the composition when a proof of A OR NOT A from Gamma is desired. (Where A is an arbitrary WFF and Gamma is any SET-OF-WFFS).

IV RELATION TO PREVIOUS WORK

This paper continues research on ESA initiated by Prof. Gerald DeJong in [4]. This work is closely related to the hybrid analytical/empirical learning methods of Mitchell et al [9], but while Mitchell's methods are restricted to learning new heuristic conditions specifying when existing operators should be applied, the generalization method described in this paper provides new problem solving operators as well as new heuristic conditions. The construction of new operators out of combinations of old ones makes our system similar to the MACROPS learning procedure of STRIPS [10] but our method is more "human oriented" and avoids reconstructing solutions during the generalization process. This is chiefly possible because we record data dependencies during problem solving. [7]. Also, STRIPS and LEX were self contained and automatic but somewhat autistic. They based learning on the results of very general (but inefficient) automatic problem solving methods whereas our emphasis is on apprentice-like systems which learn by observing the goal directed actions of efficient human experts.

V CONCLUSION

Explanation-based learning methods promise to turn examples of problem solving behavior into dramatic improvements in problem solving ability. This paper discussed generalization for improving schema-based problem solvers.

ACKNOWLEDGEMENTS

I hereby sincerely express my gratitude to Prof. Gerald DeJong for ideas, encouragement and advice. Thanks also to my office mate Jude Shavlik for improving the graphics interface and implementing a "Reason Maintenance System" in Interlisp on our XEROX 1100.

REFERENCES

[1] G. F. DeJong, "Skimming Stories in Real Time: An Experiment in Integrated Understanding," 158, Yale University Dept. of Comp. Sci., New Haven, Conn., May 1979.

[2] R. Wilensky, Planning and Understanding, Addison Wesley, Reading, Mass., 1983.

[3] G. S. Novak, "Computer Understanding of Physics Problems Stated in Natural Language," Technical Report NL-30, Department of Computer Science, University of Texas at Austin, 1976.

[4] G. DeJong, "Generalizations Based on Explanations," International Joint Conference on Artificial Intelligence-81, Vancouver, B.C., Canada, August 24-28, 1981, 67-70.

[5] G. DeJong, "Acquiring Schemata Through Understanding and Generalizing Plans," International Joint Conference on Artificial Intelligence-83, Karlsruhe, West Germany, August 8-12, 1983, 462-464.

[6] Z. Manna, Mathematical Theory of Computation, McGraw-Hill, New York, 1974.

[7] E. Charniak, C. Riesbeck and D. McDermott, "Data Dependencies," in Artificial Intelligence Programming, Lawrence Erlbaum Associates, Hillsdale, N.J., 1980, 193-226.

[8] P. O'Rorke, "Generalization for Explanation-based Schema Acquisition," Working Paper 51, Univ. of Illinois Coordinated Science Laboratory Artificial Intelligence Group, Urbana, IL, 1984.

[9] T. M. Mitchell, "Toward Combining Empirical and Analytical Methods for Inferring Heuristics," LCSR-Technical Report-27, Lab. for Computer Science Research, Rutgers: the State University of New Jersey, New Brunswick, New Jersey, March 1982.

[10] R. Fikes, P. Hart and N. Nilsson, "Learning and Executing Generalized Robot Plans," Artificial Intelligence 3, 4 (1972), 251-288.