ELSEVIER

Artificial Intelligence 112 (1999) 1–55

Artificial Intelligence

www.elsevier.com/locate/artint

Introspective multistrategy learning: On the construction of learning strategies

Michael T. Cox [a,*], Ashwin Ram [b,1]

[a] Department of Computer Science and Engineering, Wright State University, Dayton, OH 45435-0001, USA

[b] College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0280, USA

Received 25 March 1997; received in revised form 8 June 1999

Abstract

A central problem in multistrategy learning systems is the selection and sequencing of machine learning algorithms for particular situations. This is typically done by the system designer who analyzes the learning task and implements the appropriate algorithm or sequence of algorithms for that task. We propose a solution to this problem which enables an AI system with a library of machine learning algorithms to select and sequence appropriate algorithms autonomously. Furthermore, instead of relying on the system designer or user to provide a learning goal or target concept to the learning system, our method enables the system to determine its learning goals based on analysis of its successes and failures at the performance task. The method involves three steps: Given a performance failure, the learner examines a trace of its reasoning prior to the failure to diagnose what went wrong (blame assignment); given the resultant explanation of the reasoning failure, the learner posts explicitly represented learning goals to change its background knowledge (deciding what to learn); and given a set of learning goals, the learner uses nonlinear planning techniques to assemble a sequence of machine learning algorithms, represented as planning operators, to achieve the learning goals (learning-strategy construction). In support of these operations, we define the types of reasoning failures, a taxonomy of failure causes, a second-order formalism to represent reasoning traces, a taxonomy of learning goals that specify desired change to the background knowledge of a system, and a declarative task-formalism representation of learning algorithms. We present the Meta-AQUA system, an implemented multistrategy learner that operates in the domain of story understanding. Extensive empirical evaluations of Meta-AQUA show that it performs significantly better in a deliberative, planful mode than in a reflexive mode in which learning goals are ablated and, furthermore, that the arbitrary ordering of learning algorithms can lead to worse performance than no learning at all. We conclude that explicit representation and sequencing of learning goals is necessary for avoiding negative interactions between learning algorithms that can lead to less effective learning. © 1999 Elsevier Science B.V. All rights reserved.

* Corresponding author. Email: mcox+@cs.wright.edu.

[1] Email: ashwin@cc.gatech.edu.

PII: S0004-3702(99)00047-8

Keywords: Learning-strategy construction problem; Blame assignment; Learning goals; Multistrategy learning; Case-based reasoning; Meta-reasoning; Explanation; Planning

1. Introduction

The general goal of this paper is to present a theory of *Introspective Multistrategy Learning*. A multistrategy approach to learning represents a heuristic exploration of the potentially exponential space of integrations that combine multiple learning algorithms, representations, and paradigms. But until recently, this space of possibilities has remained largely unexamined and full of difficulties. Furthermore, the exploration of this space is usually done by the *researcher*, not by the *system*. The researcher decides what the system ought to learn (for example, the researcher poses a so-called "target concept") and selects the particular learning algorithm (or, in some cases, a simple sequence of learning algorithms) to be used. The goal of our research is to begin to automate this process: to develop systems that can determine their own learning goals and select and combine learning algorithms dynamically to satisfy these goals. In our view, learning in a flexible, dynamic, multistrategy learning system with these capabilities begins to resemble an active, thoughtful planning task. As with the more familiar planning in physical domains, planning in a knowledge domain involves identifying goals (here, learning goals) and sequencing operators (here, learning algorithms) into plans (here, learning strategies) to achieve these goals. The central claim of this paper, then, is that learning is fundamentally a planning task. Although this is not the first time the claim has been offered (see [40]), it is the first time that empirical evidence has been made to support the claim. We will present a computational model of learning, an implemented computer system which decides what to learn and how to learn it, and empirical performance results demonstrating that this approach provides substantial benefits over a nonplanful approach to learning. [2]

A specific focus of this research is to address the *strategy construction problem* [17] in multistrategy learning. That is, given a failure of some performance system, the computational task is to assemble a strategy to repair the problems that underlie such failures. If a learning system has access to a suite of learning methods, the task of constructing a learning strategy requires the system to select and sequence the most appropriate methods to fix the problem. A significant impediment to progress in this area is that learning algorithms have been treated as independent functions. A given learning algorithm seldom considers that other algorithms may be executed concurrently, previously, or subsequently. Yet in an unconstrained multistrategy learning system, the possibility is likely that the outcome of one algorithm will affect the results of others. For instance, if an amortization function deletes all memory elements that are at or below a given confidence threshold and a reinforcement function increments the confidence values

[2] Note that in order that the system be able to formulate meaningful learning goals, it is important for the system to have a broader performance task that the learning supports; in other words, the system must have a principled *need* to learn. The performance system we will use in this article is a story understanding system. We will use this system as the testbed for our theory of learning; however, the primary contributions of our research pertain to learning, not story understanding.

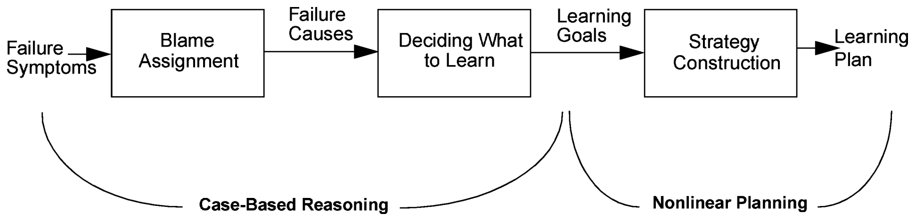

Fig. 1. Decomposition of the learning problem.

of particular memory elements, then clearly the second function call must precede the first; otherwise, any common elements at threshold will be "forgotten" and not strengthened.

As is the case with traditional planning problems, the task is to take a suitable set of learning goals and to build a learning plan having a sequence of learning steps that avoids negative interactions. This raises another crucial problem to be addressed: how are these learning goals represented? Can the system determine its learning goals, and how are those goals related to the learning algorithms that achieve them? To properly specify such a goal set, a system must explicitly decide what needs to be learned on the basis of what went wrong. In other words, the learning system must be able to understand failures in the operation of the performance system (i.e., performance failures) in causal terms related to the reasoning and knowledge of the system (i.e., reasoning failures). As with traditional diagnosis systems that have been developed to understand events in physical domains, to interpret problems, and to explain system failures, the task of generating a learning goal is to interpret and to explain events in the reasoning process (i.e., in the mental domain) and to diagnose the reasoning failure. If it is to learn effectively, the system must have knowledge about itself and a capacity to examine its own reasoning abilities. Without this willingness to introspect, we will demonstrate empirically that learning can be ineffective. In other words, a central conclusion of this research is that introspection is necessary for effective learning.

Our solution to the problem of constructing a learning strategy integrates two broad approaches in AI (see Fig. 1). Part of our research seeks to stretch the metaphor of nonlinear planning in order to fit the demands of a multistrategy learner; whereas, a second part applies the methods of case-based reasoning to understand the reasoner's faulty processes and to generate a set of learning goals. This division of responsibility determines three major subtasks. First, the learner must perform *blame assignment* (an explanatory mapping from symptom to fault in the operation of the performance system). Secondly, it must *decide what to learn* by using this explanation to form a set of learning goals (desired changes in the background knowledge of the reasoner). Thirdly, it must *construct a learning strategy* to achieve these goals by generating a learning plan (a partially ordered calling-sequence for its learning algorithms). These three tasks require that an intelligent system understand its own reasoning and learning processes well enough to learn from its mistakes.

In order to make these ideas more concrete, consider a simple example. Suppose that the story in Fig. 2 is presented to an automated story understanding system whose goal it is to interpret the story and to build a causal representation of it. If the system believes that only

Elvis was bored and asked Lynn, “Do you want to play ball?” Then Lynn went to the garage and picked up a baseball, and they went outside to play with it. She hit the ball hard so that it would reach him in leftfield. Elvis threw the ball back. They continued like this all afternoon. Because of the game, Elvis was no longer bored.

— The End —

Fig. 2. Sample input story.

children play games, then the fact that an adult such as Elvis plays ball would need to be explained to understand the story. In an attempt to explain this event and how it relates to the goals of the characters, the system may use the same knowledge it has for why children play. Subsequently, the prediction that Elvis performs this action because he wants to have fun (i.e., he has the goal to relieve his boredom) is reinforced by the explanation contained in the story. Alternatively, if the system has no knowledge of why people hit objects other than to inflict pain, the system might erroneously predict that Lynn strikes the ball in order to hurt the ball. However this explanation is contradicted by the story and represents an expectation failure of the performance system that itself must be explained in order to learn.

Yet explaining such failures and learning as a result requires that a number of difficult problems be faced. First, the system must be able to reason about the prior reasoning embedded in the story understanding task, not only to realize that the hypothesized explanation was erroneous, but to understand why that explanation was proposed in the first place. To perform such meta-level reasoning, it needs a declarative representation and explicit trace of the execution of the performance system. That is, a trace of *what* happened helps to explain *why* it happened. This trace would be a piece of knowledge that represents both the system’s expectation that the hitting event was meant to hurt (along with why the system believed this) and the actual explanation that hitting was meant to move the ball. Then to explain such reasoning, the system should realize that no alternative explanations existed and thus the proposed explanation represented a hypothesis. The issue is how to represent such traces and how to establish explanations of performance failure using such traces. Indeed, to do even this, we need a theory of what constitutes a reasoning failure and what kind of causes exist for such failures.

Secondly, to use such insights effectively, the system must identify what needs to be learned so that the failure is not repeated. It needs to form a goal to differentiate competing explanations so that they can be retrieved when appropriate. If the system has never encountered the movement explanation for hitting, it needs to have the goal to acquire it and remember it in the future. Thus after explaining the expectation failure, it must form a set of learning goals that represents specific desired changes to its knowledge base. However, this entails a nontraditional concept of goal and goal pursuit. Another set of problems, then, is how to represent learning goals, how to extract goals from explanation failures, and how to use them to guide the construction of a learning strategy.

Finally, if learning is to be cast as a planning problem, then we need to explain how a learning algorithm can be operationalized as an explicit learning operator, what the preconditions are, what the effects are, and in what formalism such operators can be realized. Explicit representation is important if the system is to reason about these algorithms in much the same manner as a planning system might reason about its actions. However, unlike the blocks-world where actions are self-evident from experience most persons share as children, it is unclear how changes to the background knowledge of a system can be represented operationally with plan schemata. Nonetheless, we will explore some possibilities using Tate's [96] Task Formalism and will demonstrate how an implemented learning system can use a nonlinear planning module to establish learning plans.

From the very early days of AI, scientists have been concerned with the issues of machine self-knowledge and introspective capabilities (e.g., [53,58]), yet few have quantitatively evaluated the trade-offs involved or investigated the nature of the role introspection assumes in learning. The research presented here uses computational introspection to assist in the choice and sequencing of learning algorithms within a multistrategy framework. But, open questions exist as to whether introspection is worth the computational overhead and in exactly what ways it facilitates the learning process. This paper begins to investigate these research questions empirically, as well as analytically. The research presented here is the first to investigate the strategy construction problem substantially, the first to examine computational introspection as a method of avoiding negative interactions, and the one to take the metaphor of learning as planning most literally (but see [40,76]).

Section 2 presents the content of an introspective theory of learning by describing a classification of reasoning failure, a taxonomy of failure causes, a typology of learning goals, and a representation with which to express reasoning failure. Section 3 describes the Meta-AQUA multistrategy learning system that operates in the domain of story-understanding failures. This section includes the process divisions and algorithms that implement our learning theory. Meta-AQUA incorporates a story generation module by which experimental trials are generated, a performance module that understands these stories, and a multistrategy learning module which improves the performance task. Section 4 presents two extended examples to illustrate our learning method and to clarify the evaluation that follows. Section 5 provides a computational evaluation of the assertion that learning must include a phase that explicitly decides what to learn through an introspective mechanism. We show that without a focus for learning as provided by a set of deliberate learning goals, learning may not only be less efficient, but it may be worse than no learning at all. Finally, Section 6 presents related research and summarizes our contributions.

2. Representation and theory

Failure provides both human and artificial reasoners with strong clues when deciding what needs to be learned [1,20,30,35,37,41,44,67,87,90,93,95]. One of the major goals of establishing a theory of introspective learning, therefore, is to provide both a general characterization of reasoning failure and the potential causes of such failure in order to discover the nature of these clues. A sufficient characterization of failure will categorize the kinds of cognitively salient symptoms that signal to the reasoner that something worth learning exists. A sufficient taxonomy of the causes of failure will include those factors that account for each symptom in enough detail as to enable learning from them. The learner's task, then, is to perform an explanatory mapping from symptom to fault, and thus, to determine what causes a particular failure. Such explanations detail what needs to be learned by circumscribing the faults that must be corrected.[3]

In our model, reasoning is performed upon the representation of some input. The input is not just perceived, but in addition, an attention mechanism filters the input as determined by the current mental state of the reasoner. The critical elements of the reasoner's mental state are the goals and the expectations present in the reasoner's memory. The filtered input and the reasoner's knowledge, goals, and expectations then determine some interpretation of the input, thus eliciting additional goals and expectations. These conditions present a rich context from which to detect a failure.

Besides providing an inferential focus for the reasoner, a fundamental purpose of forming expectations is to test the general limits of knowledge independent of the particular goals of the moment. That is, agents generate expectations to improve the boundaries of their knowledge: to retract those parts of the boundaries that are incorrectly extended and to expand the limits where gaps exist. An expectation represents a hypothesis or projection of current knowledge that is falsifiable. One of the most basic mental functions, therefore, is to compare one's expectations with environmental feedback (or, alternatively, a "mental check" of conclusions) to detect when the potential for improvement exists. That is, the reasoner explicitly calculates some expected outcome and compares it with the actual outcome that constitutes the feedback.

An *outcome* is defined broadly without reference to a specific task. The *expected outcome* could be the result of either a problem-solving process or a comprehension process such as story understanding. Comprehension processes attempt to predict and understand events in a stream of input. Therefore, the outcome would be an interpretive understanding of a system's input, such as a reader's comprehension of successive textual sentences. Text can violate the reader's tacit expectations concerning what will be observed. In the short story of Fig. 2 for instance, the system implicitly expects people to hit animate objects, even though it did not generate that expectation prior to encountering the sentence containing a person that struck the ball. To satisfy the comprehension task in complicated or unusual input, questions may be raised and an explanatory process may be invoked. Hence, the expected outcome will be an explanation [72,74,88,99].[4] That is, the reasoner consciously anticipates a certain explanation to be true of some object or event

[3] This paper's position does not claim that all learning is guided by failure. Success contributes to learning as well, but the impetus for learning resides entirely with failure in the theory of learning presented here. See [20], however, for a computational argument for why failure may be preferred over success in learning.

[4] The relationship between question asking and explanation is not obvious. Sometimes an expected outcome is described as an explanation of an anomaly, while at other times an outcome is described as the answer to a question. The relationship is that anomalies cause questions to be posed of the form "Why did some event occur in the input?" The answer is an explanation that answers this question. In addition, the specific mental process that forms an expectation (expected outcome) is not determined a priori. The process may be either an inferential process such as deduction, or it may be a memory process that retrieves an expectation from memory. For instance, to understand a story input, the reasoner (reader) may retrieve from memory a schema with which to interpret the story fragment.

in the input that is to be explained. When these explanations prove incorrect, such as the explanation that Lynn wanted to hurt the ball, explicit expectations can be violated as well.

Finally, the *actual outcome* may originate internally or externally. Feedback may come from the environment (via perceptual/interpretive processes), or, it may emanate from a mental process such as an arithmetic check of a mathematical computation. In all such cases, the actual outcome is compared with the expected outcome in order to decide whether or not a failure exists in reasoning. If such a failure is detected, the reasoner attempts to explain the failure and to learn from it.

2.1. Reasoning failures and failure causes

A reasoning failure is defined as an outcome other than what is expected (or a lack of some outcome or appropriate expectation). Given the above model of outcome comparison, Table 1 presents a logical matrix of failure symptoms that depend on the outcome values. The expected outcome may or may not have been produced; thus, the expected outcome node, E, either exists or does not exist. Also, the actual outcome node, A, may be present or it may not.

Expected outcomes are confirmed when they match the actual outcomes and reasoning proceeds smoothly. However given a mismatch, two types of failure can result. A *contradiction* occurs when a positive expectation conflicts with the actual result of either reasoning or action in the world. An *unexpected success* occurs when the reasoner did not believe that reasoning would be successful, yet it was nonetheless. Alternatively, failure may happen when no expectation is generated prior to some outcome. That is, an *impasse* occurs when the reasoner cannot produce a solution or understanding prior to being given it; whereas, a *surprise* occurs when an actual outcome demonstrates that the reasoner should have attempted a solution or prediction, but did not. Finally, a *false expectation* is the case where a reasoner expects some positive event, but none occurs (or when a solution is attempted for a problem having no solution). The *degenerate* case has no potential for learning and represents the condition such that no expectation was generated and no outcome presents itself.

Each of these failures may be caused by many factors. Although a general solution to the problem of blame assignment does not exist, we may frame the potential causes of failure so as to better understand the space of blame assignment. Clearly, reasoning is intentional and oriented toward the pursuit of specific *goal states*. Moreover, we assume that reasoning

Table 1
Taxonomy of failure symptoms

	$\exists E$ (expectation exists)	$\nexists E$ (expectation does not exist)
$\exists A$ (actual exists)	Contradiction Unexpected success	Impasse Surprise
$\nexists A$ (actual does not exist)	False expectation	Degenerate (N/A)

Table 2
Taxonomy of causes of reasoning failure

	Knowledge states		Goal states		Processes		Environment	
	Domain knowledge	Knowledge selection	Goal generation	Goal selection	Processing strategy	Strategy selection	Input	Input selection
Absent	Novel situation	Missing association	Missing goal	Forgotten goal	Missing behavior	Missing heuristic	Missing input	Missing context
Wrong	Incorrect domain knowledge	Erroneous association	Poor goal	Poor selection	Flawed behavior	Flawed heuristic	Noise	Incorrect context
Right	Correct knowledge	Correct association	Correct goal	Correct association	Correct behavior	Correct choice	Correct input	Correct context
	Theory	Memory	Desires	Opportunity	Action	Control	Perception	Attention

uses *knowledge* to *process* perceived input from some *environment* in order to create a representational state of the world and to achieve these desired goals. Reasoning processes transform specific mental states into new states. Some of these states are knowledge states representing facts and experience, some are perceived states representing conditions in the environment, and some are goal states representing desired new states. Based upon such representations, decisions result in actions that change the world, thus producing new environmental states that can subsequently be input or perceived in order to compare the goal to the actual state of affairs in the world. Such decisions result in new internal actions that may change the expectations present in working memory that bias later input. Given these assumptions, reasoning will fail if any of the constituents of reasoning fail; that is, if a problem exists with the reasoner's knowledge, goals, mental processes, or input from the environment.

In addition, not only can these components be a likely cause of error, but the ways in which the reasoner *selects* them can also be a source of error. Nonselection is an important and often overlooked factor in the analysis of failure. It is a result of poor memory organization rather than incorrect memory content. Failure can occur, not because an agent does not know some fact, but because the agent cannot retrieve the fact when needed [18].

Table 2 presents a matrix for summarizing the causal factors that bear on the determination of blame. As indicated at the heading in the uppermost row, the table is divided into the four major causal categories. Failure could stem from the knowledge states with which the reasoner makes decisions, goal states generated during reasoning, the reasoning processes used to achieve the goals, or the input that represents the environment and from which feedback is provided. In each of these categories, the relevant item may be either missing or wrong. Omission errors occur when a necessary component is not present (this is represented by the "Absent" row in the table); whereas, commission errors occur when an incorrect component is present (this is represented by the "Wrong" row in the table). In addition, because knowledge is embedded in a memory and must be retrieved before it can be used to pursue a goal, an error of omission can result from

Unary Goals	**N-ary Goals**
• Knowledge Acquisition Goal	• Knowledge Differentiation Goal
• Knowledge Refinement Goal	• Knowledge Reconciliation Goal
• Knowledge Expansion Goal	• Knowledge Organization Goal
• Knowledge Organization Goal	

Fig. 3. Taxonomy of learning goals.

nonselection, rather than simply nonexistence. If an item is correct, then that category contributes nothing to the failure. A characterization of each factor is present in the lower row of the table.

2.2. Explicit learning goals

To provide focus for learning, a set of learning goals are spawned from an explanation of the failure. Just as standard goals represent what an agent needs in the world, *learning goals* represent what a system needs to know [20,72,73,76,78]. These goals are spawned when deciding what to learn or when subgoaling on a superordinate learning goal. Learning goals help guide the learning process by suggesting strategies that would allow the system to learn knowledge required to avoid future failures. Learning goals specify the desired structure and content of knowledge, as well as the ways in which knowledge is organized in memory. Learning goals also facilitate opportunistic learning (see [36,72,73,76]); that is, if all information necessary for learning is not available at the time it is determined what is needed to be learned (e.g., when a question is posed), then a learning goal can be suspended, indexed in memory, and resumed at a later time when the information becomes available.

Fig. 3 lists the types of learning goals used in our theory. All of these goals are achievement goals because they attempt to achieve some new state in the background knowledge, rather than prevent or maintain some state.[5] Some learning goals seek to add, delete, generalize or specialize a given concept or procedure. Others deal with the ontology of the knowledge (i.e., with the kinds of categories that constitute particular concepts). Many learning goals are unary in that they take a single target as argument. For example, a *knowledge acquisition goal* seeks to determine a single piece of missing knowledge, such as the answer to a particular question. A *knowledge refinement goal* seeks a more specialized interpretation for a given concept in memory, whereas a *knowledge expansion goal* seeks a broader interpretation that explores connections with related concepts.

Other learning goals are n-ary, taking two or more arguments. A *knowledge differentiation goal* is a goal to determine a change in a body of knowledge such that two or more items are separated conceptually. In contrast, a *knowledge reconciliation goal* is one that seeks to merge multiple items that were mistakenly considered separate entities.

[5] Of course, learning goals (or policies) that either prevent the removal of a strongly held belief or maintain some special mental state are conceivable, but outside the scope of this research.

Both expansion goals and reconciliation goals may spawn a *knowledge organization goal* that seeks to reach a particular configuration of the background knowledge. Usually a learner wants to reorganize the existing knowledge so that it is made available to the reasoner at the appropriate time, as well as modify the structure or content of a concept itself. Such reorganization of knowledge affects the conditions under which a particular piece of knowledge is retrieved or the kinds of indexes associated with an item in memory. Because the desire can be either to organize a particular state or to organize two or more states with respect to each other, this goal type may take one or more arguments.

2.3. *Representation of reasoning failure*

An early tenet of artificial intelligence is that reasoning about the world is facilitated by declarative knowledge structures that represent salient aspects of the world (e.g., [2,53,63, 70,85]). An intelligent system can better understand and operate in such a represented world as opposed to one in which knowledge is encoded procedurally or implicitly. The system may inspect and manipulate such structures, the system can be more easily modified and maintained, and such representations provide computational uniformity. But if a system is to reason about itself, this principle can be applied equally to representations of its own reasoning and knowledge. Given such structures, a system can reason about its own memory system when it forgets and can reason about its knowledge and inferences when it draws faulty conclusions.

2.3.1. *Representing contradiction: A symptom of failure*

We posit a declarative representation for mental state transformations using *explanation pattern* (XP) theory [73,74,88,89,91] and call the representational structures *meta-explanation patterns* (Meta-XPs). Because meta-X literally means "X about X", a meta-explanation pattern is an explanation pattern about another explanation pattern. While a standard XP is a causal structure that explains a physical state by presenting the prior chain of physical events influencing such states, a Meta-XP, such as the one illustrated in Fig. 4, is an explanation of how or why an XP is incorrectly generated or otherwise fails.[6] Two classes of Meta-XPs exist to facilitate a system's ability to reason about itself and to assist in constructing a learning strategy. A *Trace Meta-XP* (TMXP) explains *how* a system generates an explanation about the world (or itself), and an *Introspective Meta-XP* (IMXP) explains *why* the reasoning captured in a TMXP goes awry.

TMXPs record the extent of reasoning tasks and the reasons for decisions taken during processing. IMXPs are general causal structures that represent explanations of the failure types listed in Table 2. Whereas a TMXP records the immediate mental events of the reasoner during story understanding, an IMXP is retrieved from memory and applied to the TMXP to support problem diagnosis. As such, the TMXP captures symptoms of failure and the IMXP captures the fault. This case-based approach to self-understanding is similar to the case-based operations by which standard XPs are retrieved and applied to input representations during story understanding. The same basic algorithm [74] is used in both.

[6] Here the definition of a Meta-Explanation is interpreted in a narrow sense as applied to understanding tasks involving the explanation of anomalies. In general, however, a Meta-XP may be any explanation of how and why an agent reasons in any particular way, including processes other than explanation.

Like XPs, Meta-XPs are directed graphs with nodes that are either states or processes and links that are either ENABLES links (connecting states with the processes for which they are preconditions), RESULTS links (connecting a process with a result), or INITIATE links (connecting two states). The XP provides an internal cause (causal justification) for a distinguished node called the EXPLAINS node by providing its causal antecedents. In the graph, all sink nodes are called PRE-XP-NODES and represent the applicability conditions of the explanation. All source nodes are XP-ASSERTED-NODES and represent the terms with which to evaluate the explanation. If the set is completely believed, the explanation is accepted, if one or more in the set is not believed, then explanation is rejected, and if a member of the set is unknown, then the node is recursively explained. All other nodes in an XP are INTERNAL-NODES.

Nodes in a Meta-XP represent mental states and mental actions (processes). In our ontology, a nondetermined mental process is labeled a *Cognize* node. It can be specialized to either an inference process, a memory retrieval process, or an I/O process.[7] Therefore, an intelligent agent can respectively form an expectation by inferential reasoning (logical or otherwise), by remembering, or via another agent's communication. The basic organization for all representations of failure is at the level of a comparison between an expectation and some feedback (either from the environment or other inference).

As an example of a Meta-XP representation for an entry in Table 1, Fig. 4 illustrates the basic structure of a contradiction.[8] Some goal, G, and context or cues, C, enables some cognitive process to produce an expected outcome, E. A subsequent cognitive mechanism produces an actual outcome, A, that, when compared to E, fails to meet the expectation. This inequality of actual outcome with expected outcome initiates the knowledge of contradiction. If the right-most *Cognize* node was some inferential process, then the failure becomes an expectation failure and the node C represents the context, whereas if the process was a memory function, the contradiction is called an incorporation failure and C represents memory cues.

In Fig. 4 the PRE-XP-NODES are A, E, and the node labeled "Expectation or Incorporation Failure". This latter node is the EXPLAINS node (i.e., the one that the Meta-XP explains). The left-most Cognize node, the Compare node, and the nodes G and C form the XP-ASSERTED-NODES set. All others are internal nodes. Such representations allow a system to reason about various causes of failure (e.g., a poor goal or a missing item in the context), although in the form shown we have simply represented a class of symptoms. No specification is actually provided here that determines what the fault or cause of the failure is. In other words, Fig. 4 illustrates an item from Table 1 rather than Table 2. The cause remains to be inferred.

The terms used to represent reasoning failure represent the vocabulary labels that compose meta-explanations. We propose two types of commission error labels. *Inferential expectation failures* typify errors of projection. They occur when the reasoner expects an

[7] This is in keeping with Schwanenflügel et al. [92] who analyzed folk theories of knowing. Subject responses during a similarity judgement task decomposed into inference, memory, and I/O clusters through factor analysis.

[8] Numbers on the causal links (thick arrows) indicate relative temporal sequence [19] and the prefix *mentally* on causal link labels refers to internal rather than external effects and preconditions. Attributes and relations are represented explicitly in these graphs. For instance, the ACTOR attribute of an event X with some value Y is equivalent to the relation `ACTOR` having `domain` X and `co-domain` Y.

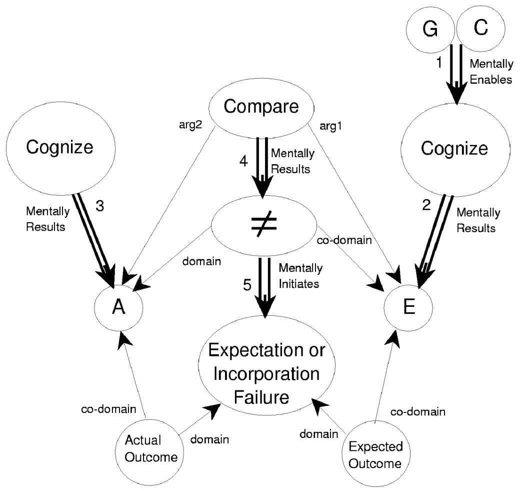

Fig. 4. Meta-XP representation of contradiction. A = actual; E = expected; G = goal; C = context or cues.

event to happen in a certain way, but the actual event is different or missing. *Incorporation failures* result from an object or event having some attribute that contradicts some restriction on its values. These two are similar except that the expectation is explicit in the former and implicit in the latter. In addition, we propose four omission error labels. *Belated prediction* occurs after the fact. Some prediction that should have occurred did not, but only in hindsight is this observation made. *Retrieval failures* occur when a reasoner cannot remember an appropriate piece of knowledge; in effect, it represents a memory failure. *Construction failure* is similar, but occurs when a reasoner cannot infer or construct a solution to a problem. *Input failure* is error due to lack of some input information. Combinations of these are used to represent each reasoning failure type (symptom) listed in Table 1.

2.3.2. Representing forgetting: An introspective explanation of failure

Given a characterization of reasoning failure as symptom taken from our error taxonomy, the task of blame assignment is to explain the failure in terms of why it occurred. This diagnosis task then is to map the failure symptoms to combinations of faults taken from Table 2. In addition to representing a trace of the reasoning in a TMXP, a learning system that performs this task needs a library of IMXPs that cover the range of errors possible in the performance task. But the representation for these causal patterns of failure are not always straightforward. For instance, consider the difficulties of representing a failure due to forgetting.

Because forgetting is not a mental event, but rather the lack of successful memory processing, challenges exist when representing it.[9] Forgetting can be expressed properly only if a system can represent that it does not believe a successful memory retrieval has occurred. The belief logic of Doyle [28] has four truth values for a given proposition “*p*”.

[9] See [14] for alternative representations and the associated problems they entail.

If *p* is believed then it is "in" the set of beliefs, whereas if *p* is not believed then it is "out". Conversely, the negation of the assertion of *p* may be either in or out of the agent's set of beliefs. Therefore, the four truth values are $p(\texttt{in})$, $p(\texttt{out})$, $\neg p(\texttt{in})$, and $\neg p(\texttt{out})$. Using these values, a system needs to be able to declare that there is a memory item that was not retrieved.

The system could create a dummy concept, *d*, representing the forgotten item that it believes did not result from some retrieval process. This concept could be marked as not believed (i.e., $d(\texttt{out})$), since it was not retrieved and cannot be specified by the system. But technically, it is incorrect to assert that the concept is not believed if it is in the system's long-term memory (the *background knowledge* or *BK*). In other words, it may be believed but not recalled. Our response to this dilemma is first, to assume a special set of beliefs representing the working memory of the system (the *foreground knowledge* or *FK*), and then secondly, to modify Doyle's belief logic to claim belief membership with respect to a particular set of beliefs. Thus, *m*, a given memory item that was not retrieved, may be in the set of beliefs with respect to the BK, written $m(\texttt{in}_{\texttt{BK}})$, but out of the set of beliefs with respect to the FK, written $m(\texttt{out}_{\texttt{FK}})$. [10]

The Meta-XP structure of Fig. 5 represents a memory retrieval attempt enabled by goal, G, and cues, C, that tried to retrieve some memory object, M, given an index, I, that did not result in an expectation (or interpretation), E, that should have been equal to some actual item, A. The fact that E is out of the set of beliefs with respect to the reasoner's foreground knowledge (i.e., is not present in working memory) initiates the knowledge that the symptom of impasse (specifically a retrieval failure) had occurred.

This structure can represent an entire class of memory failures: failure due to a missing index, I; failure due to a missing object, M; failure because of a missing retrieval goal, G; [11] or failure due to not attending to the proper cues, C, in the environment. These alternatives are disambiguated by the annotation of $\mathrm{M}(\texttt{out}_{\texttt{BK}})$ attached to the truth value of M. The annotation explains the forgetting by postulating that no memory item exists in the BK to retrieve. [12] Although it may be the case that another cause (e.g., a missing index) may have caused the error, it allows the system to begin the mapping from symptom to fault at this point when trying to determine a goal to learn.

A list of specific learning-goals that need to be spawned are included as part of the representational structure of each IMXP. When the IMXP is bound to the trace of the failure, the goals are automatically bound to particular points in the representation of the trace (via unification of variables) that provide a possible location of failure. In the case of a missing node *M*, a goal is attached to acquire the missing knowledge. As will be seen in the extended example from Section 4.2, these are only starting points; additional subgoals

[10] Compare this with the assumption maintenance system discussed by McDermott [55]. In general, propositions may be `in` or `out` with respect to arbitrary sets of beliefs, which in the Meta-AQUA system are used to represent what is in the FK during different reasoning experiences.

[11] A missing retrieval goal occurs when the agent never attempted to remember. For instance, the reasoner may have wanted to ask a question after a lecture was complete, but failed to do so because he never generated a goal to remember. Alternatively the agent may know at the end of the lecture that he needs to ask something, but cannot remember what it was. This second example is the case of a missing index.

[12] If M is not present in the BK, functionally it does not matter whether or not *M* was in memory at a previous time.

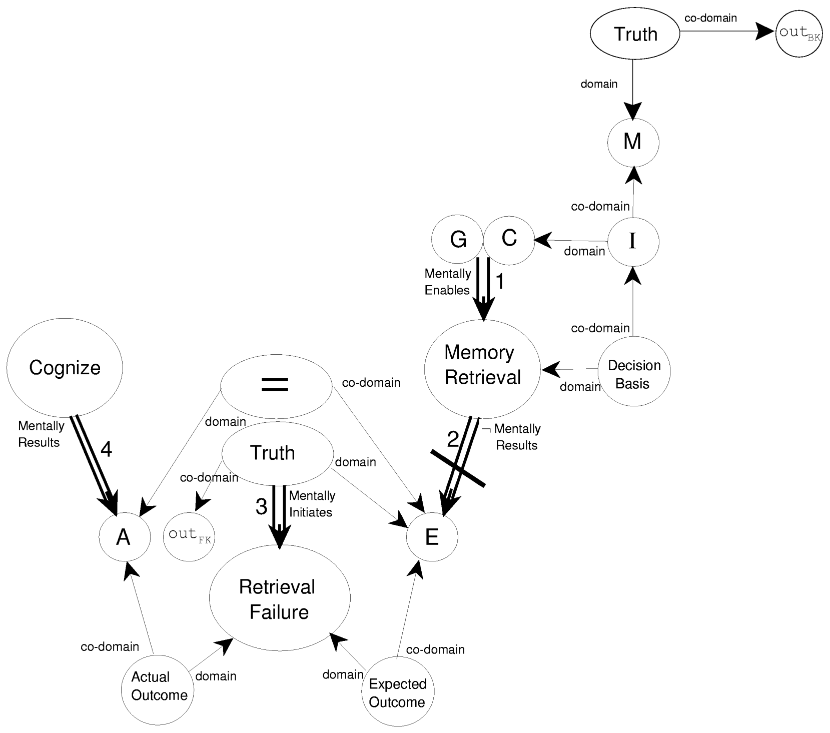

Fig. 5. Meta-XP representation of forgetting (impasse). A = actual; E = expected; G = goal; C = cues; M = memory item; I = memory index.

may be spawned to support the original goals from the IMXP or learning goals can be abandoned in the face of changing circumstances (e.g., if it is discovered that M really does exist).

We have developed similar representations for the other causal explanations enumerated in Table 2 [13]. The second-order formalism outlined here not only supports the construction of a learning strategy, but it also provides a knowledge framework with which a system can represent knowledge of itself. This can be useful for a system that must communicate its reasoning to the user of a system [11] and for a system that must infer the mental events and states of others (see [54,69]; but see also Reilly and Bates [82] who use a minimalist attitude and Zeng and Sycara [101] who use no explicit second-order structures to make such inference).

3. Process divisions of the Meta-AQUA system

Meta-AQUA is a goal-driven learning system that chooses and combines multiple learning methods from a toolbox of algorithms in order to repair faulty components responsible for failures encountered during the system's performance task. The performance task is subjective story understanding [4,22,50,62,72,99]. System input is in the form of a stream

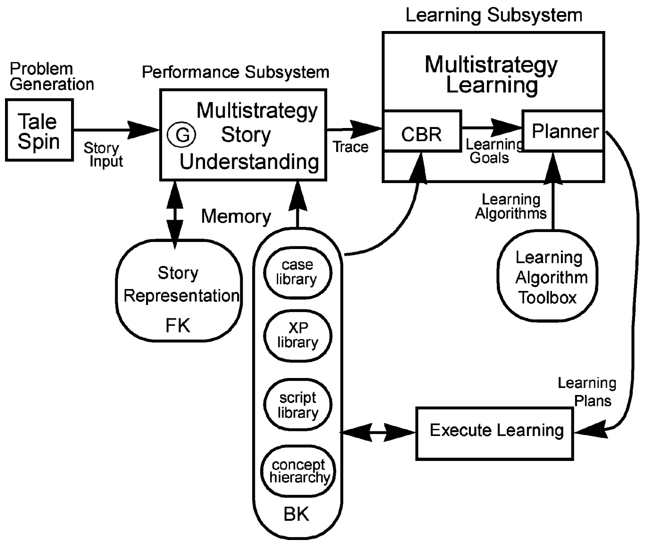

Fig. 6. Detailed Meta-AQUA system architecture.

of conceptual entities representing events in a story, and the performance task is to create a coherent explanatory and predictive model of the interactions between characters and the events. But when a story understander predicts that a particular explanation will hold in a story and an alternative explanation is subsequently given in the story, then a performance failure has occurred. To learn from this kind of failure, the system must understand why the failure occurred, must form a goal to learn from the failure, and must create a learning strategy to change the knowledge that caused the failure.

The Meta-AQUA architecture and flow of information within the system is shown in Fig. 6. The problem generation module outputs a story to the story understanding performance system with the initial goal to understand the input. The performance module uses schemas from its background knowledge (BK), represented using combinations of XPs and scripts, to explain the story and to build a representation for it in its foreground knowledge (FK). When this task fails, a trace of the reasoning that preceded the failure is passed to the learning subsystem. A case-based reasoning [5,34,45,83,87] subsystem within the learner uses past cases of introspective reasoning from the BK to explain the comprehension failure and to generate a set of learning goals. These goals, along with the trace, are then passed to a nonlinear planner [84,96,100]. The planner subsequently builds a learning strategy from its toolbox of learning methods. The learning plan is passed to an execution system that examines and changes items in the BK. These changes enable improved future performance.

Meta-AQUA is programmed in Symbolics Common LISP under the Genera operating system (Version 8.3). The hardware platform is a Symbolics MacIvory Model-3 LISP microprocessor embedded in a Macintosh IIci personal computer. Including comments and documentation, the LISP source code takes up approximately 750 kilobytes of disk space in sixty-seven files.

Elvis was bored. Elvis asked Lynn, “Would you push the ball2 to me away from you?” Lynn went to the garage. She picked up the ball2. She had the ball2. She went to outside. He went to outside. He played with the ball2. She hit the ball2. She hit the ball2 because she wanted to move the ball2 to him. He hit the ball2. He hit the ball2 because he wanted to move the ball2 to her. He played with the ball2 because he didn’t want to be bored.

— The End —

Fig. 7. Tale-Spin story TS1.

3.1. *The input: Elvis-World and Tale-Spin*

To support large data collection for empirical evaluation, the Tale-Spin story generation program[13] provides a potentially infinite number of input variations that test Meta-AQUA’s ability to learn from explanation failure. Given a main character and a problem, Tale-Spin simulates the actions that would be necessary for the character to achieve goals stemming from the problem. For instance, if a character is bored, Tale-Spin assigns the character an initial goal to remove the state of boredom. The character can achieve the goal by convincing a friend to play, finding a ball, going outside, and then batting the ball back and forth (as with the story in Fig. 7).[14] For each event in the story the generator adds any associated causal results (along with occasional random events and states). These results change the world and enable further actions by characters in the story. For example, the act of getting the ball and going outside enables the hitting of the ball which results in the ball’s movement between the characters. In turn, these actions remove the boredom. Tale-Spin terminates a story when the goals and subgoals of the main character have been achieved or when all possible actions to achieve them have been exhausted.

Among the changes to Tale-Spin, we added a musician named Elvis and a police officer to the cast of characters. Elvis is temporarily boarding with Mom, Dad and their daughter Lynn, whereas the officer occasionally visits the house, presumably because of neighborhood complaints of loud music and raucous behavior. Furthermore, the police officer often (but not always) brings a drug-detection dog along with him. We also added two new problem types to the original problems of thirst and boredom. Characters may now be *jonesing*[15] for drugs. In Elvis’ case, he sometimes smokes marijuana to relieve his jones, whereas Dad and Lynn occasionally smoke tobacco. The police officer has the problem of being *concerned* about the law. This problem is solved if he can either locate contraband or arrest criminals. We modified Tale-Spin to include in the output

[13] Tale-Spin [56] was obtained from the UC Irvine repository. Pazzani [68] used it to evaluate the OCCAM multistrategy learning system.

[14] Note the correspondence to the sample story from Fig. 2. Unlike the first story, Tale-Spin generates stories that are both stylized and simple. It also provides random events and all necessary explanations. If the research focus was on the performance task, Meta-AQUA would process unconstrained text and the burden of explanation verification would be upon the inference capability of the performance system.

[15] In the vernacular, a “jones” is a drug habit accompanied by withdrawal symptoms. The verb “to jones” is to be going through a state of withdrawal.

explanations associated with each problem of the main characters (usually, that a character performs actions because the character has the goal that is the outcome of the action). We also reprogrammed Tale-Spin to hide the marijuana during story initialization in different locations (e.g., in the cupboard, refrigerator, and under the carpet), so the officer's task varies depending on entry conditions (i.e., at what point in the story the officer arrives on the scene and whether the dog accompanies him), the initial location of the pot, and the actions of the characters in the story.

3.2. The performance task: Story understanding and explanation

Understanding involves building causal explanations of an input, whether that input is a visual scene, spoken language, or written text. These explanations provide conceptual coherence [38,39,66,80] by incorporating the current input into pieces of the previous input and by generating expectations about subsequent input. The understander skims a stream of input by instantiating schemas to fit each input item and linking it into the model of previous input, unless the current input is anomalous or unusual. If an anomalous situation is identified, then the understander must explain the input by elaborating it beyond simple schema instantiation. For an agent to achieve a comprehension goal in story understanding, it must be able to meaningfully link the current input to both the preceding and forthcoming events in the story.

The basic task of story understanding is shown in Fig. 8 and represents a multistrategy variant of the AQUA story-understanding algorithm [72,74]. Given some input and a current context (including a comprehension goal, the system's BK, and within the FK_1, a current model of the previous input), if the input is interesting, [16] the system's task is to choose or construct a strategy with which to explain the input (e.g., case-based reasoning or explanation pattern application), otherwise it must incorporate the input into FK_1. Upon execution of the explanation strategy, the system outputs a new representation (FK_2) of the input that has no anomaly and is coherent with respect to the BK. The input is understood given that it remains consistent and coherent in the face of future input. To support the learning task if failure results, the system outputs a representational trace of the reasoning that produced the understanding.

The explanation of interesting input should further the overall goal of understanding the entire story. As already mentioned, the explanation is a good one if it helps to incorporate the new input with previous input and it needs little or no re-explanation when given further input concerning the same topic. The explanation is also good if it addresses the particular features that initially made it interesting [74,77].

As detailed in [22], three processes exist in the understanding task used to process stories. First, the understander needs to identify anomalous (or otherwise interesting) input. In the absence of interesting story passages, the reader skims the input by passing it

[16] Interesting input is either an anomalous conceptualization or something pertaining to the intrinsic goal of the reasoner. For example, sex, violence, and loud noises are intrinsically interesting [86]. In addition, anything concerning a concept about which something has been learned recently will be categorized as interesting [13]. For a more detailed set of interestingness heuristics see [71].

```
Input:
  • Comprehension goal
  • Input
  • BK (includes library of explanation strategies)
  • $FK_1$
If the input is interesting then
  Construct or select an explanation strategy to elaborate the input
  Execute strategy
  Incorporate story input into $FK_1$
else skim the input.
Output:
  • $FK_2$
  • Trace
The input is understood if, given future input, $FK_2$ remains consistent
and coherent.
```

Fig. 8. Understanding specification.

to a simplified version of SAM, a script application program [24,25].[17] Second, given interesting input the reader generates a hypothetical explanation to explain the text. Third, it verifies the generated explanation. Both explanation generation and verification involve strategy construction (or selection). The understander must construct (or select) a method to generate an explanation and to construct (or select) a method to test the veracity of the explanation.

Assuming such a model for the story understanding performance task, traces of system performance can be specified and recorded at run-time in declarative structures. TMXP reasoning traces are used by learning mechanisms to reason about processing failures, if and when failure occurs. A TMXP contains a *decide-compute node* (D-C-NODE) for each of the sub-processes of an understanding task; that is, it records the decision and the justifications behind each decision (see Fig. 9). Both the generation and verification processes have four steps each of which correspond to a process field in a D-C-NODE. The four fields are input analysis, goal specification, strategy decision, and strategy execution. For each field, the record stores both the enabling conditions and the resulting state. For the first three fields, the D-C-NODE records the decision basis (i.e., the knowledge, heuristics or inferences that justify the decision choice), and for the last field, it records the side-effects of the process. Full details are presented in [13].

[17] The script applier understands a story by matching input sentences to stereotypical sequences of events (i.e., to scripts). For example, a simple *pipe-smoking* script consists of subscenes to get the pipe, put tobacco into it, smoke it, then clean it, and hierarchically, these scenes are composed of subscenes (see [22] for a full script representation). Although scripts omit many of the causal relations between events in a story, they can help an understander interpret a story by providing details not explicitly mentioned in the story.

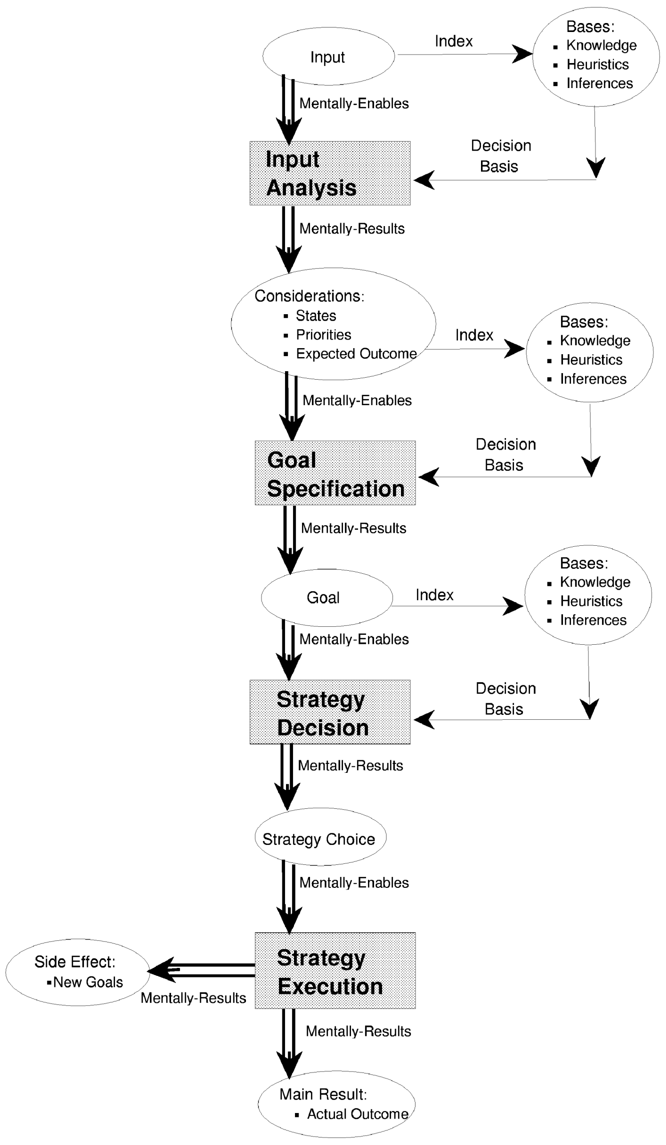

Fig. 9. Graph structure for decide-compute Node.

If a failure occurs (as detected by the algorithm to be presented in the forthcoming section), the system suspends the understanding performance task and invokes the learning task. When this happens, the trace of the reasoning along with a characterization of the failure (a symptom as determined by the failure detection algorithm) is passed to the learning process for introspective explanation. When learning abates, the system resumes the story understanding performance task.

3.3. The learning task: Case-based introspection and nonlinear planning

The Meta-AQUA system learns about drug-smuggling and simple sports activities, given its prior experience with stories about terrorists and its general knowledge of physical causality. When the Meta-AQUA system detects an explanation failure, the performance module passes a trace of the reasoning (i.e., a TMXP) to the learning subsystem. At this time, the learner needs to explain why the failure occurred (assign blame) by applying an introspective explanation to the TMXP. An IMXP is then retrieved using the failure symptom as a probe into memory. Meta-AQUA instantiates the retrieved meta-explanation and binds it to the trace of reasoning that preceded the failure. The resulting structure is then checked for applicability. If the IMXP does not apply correctly, then another probe is attempted. An accepted IMXP either provides a set of learning goals (determines what to learn) that are designed to modify the system's BK or generates additional questions to be posed about the failure. Once a set of learning goals is posted, they are passed to the nonlinear planner for building a learning plan (learning-strategy construction).

During the processing of stories such as these, Meta-AQUA records its reasoning in a trace structure as described above so that it can pass relevant information to the learner upon failure. These knowledge structures contain representations for each of the reasoning sub-processes: interest identification, explanation formation, and verification. For each, the structure records the considerations that prompted the process, the bases for making a reasoning strategy decision, and the result of strategy execution. Using information from the trace, learning is divided into three sub-processes: failure identification, learning generation, and verification.

The first process performs failure detection. Five possible types of failures as listed in Table 2 can occur. Failure detection inputs two structures (an expected outcome, E, and the actual outcome, A) and the trace of the reasoning producing these knowledge structures. The algorithm for this process is shown in Fig. 10. The detection process occurs either during the verification phase of the performance task of the system or during the generation phase after a resumption of a suspended generation goal. This second condition occurs after the performance system previously tried to generate a hypothesis, but could not. The generation phase suspends the goal and new input later provides the answer (see impasse condition in Fig. 10). Along with the trace, the process outputs a determination of which of the failures exist (if any) to the next phase.

The second phase concerns the actual determination of the causes of failure and the construction of a learning strategy which is then executed. Fig. 11 defines this learning task and shows the overall information flow to and from the learning process. [18] The input to learning is a general goal to learn from the failure, a trace of the prior reasoning, the story model in the FK, and the BK. The output of the phase is an implicit hypothesis that the learning was correct along with an augmented trace. The changes to the BK from learning are attached to a set of D-C-NODEs and are indexed in memory where the changes occur. Fig. 12 outlines the algorithm that computes the changes in the second phase. The algorithm will be explained in detail below.

[18] Cox and Ram [22] emphasize the similarity between story understanding and learning. Compare Fig. 11 with Fig. 8.

```
Detect-failure (E, A, trace)
If (A ($out_{FK}$) and trace indicates time to event expired)
or (A ($out_{FK}$) and Is-impossible (goal (generate, E))) then
    return false expectation
If E($in_{FK}$) then
    If E $\neq$ A then
        return contradiction
    else if E = A then
            If expected to fail then
                return unexpected success
            else return success
else if $\exists$ goal (generate, E) then
        return impasse
    else return surprise
```

Fig. 10. Failure detection algorithm.

Input:

- Learning Goal
- Reasoning Trace
- FK
- BK_1

If a failure occurred during the trace then

Construct a learning strategy to repair BK_1
Execute strategy
Store reasoning trace

Output:

- BK_2

The knowledge is repaired if, given a similar future situation, the failure will not recur

Fig. 11. Learning generation specification.

The third phase concerns verification. Although beyond the scope of this paper and more suitable for future research, verifying the learning could involve either of two strategies. The system could be reminded of a change to the BK (as associated with the D-C-NODEs and described above) at some future time when the changed knowledge is reused. The learning can then be checked as to whether it is effective. Alternatively, the system could

```
0. Perform and Record Reasoning in TMXP Trace
1. Failure Detection on TMXP
2. If Failure Then
     Learn from Mistake:
       • 2 a. Blame Assignment
           Compute index as characterization of failure
           Retrieve Introspective Meta-XP
           Apply IMXP to TMXP
           If Meta-XP application is successful then
              Check IMXP antecedents
              If one or more nodes not believed then
                 Introspective questioning
                 GOTO step 0
           Else GOTO step 0
       • 2 b. Create Learning Goals
           Compute tentative goal priorities
       • 2 c. Choose Learning Algorithm(s)
           Translate TMXP and goals to predicates
           Pass predicates to planner (Nonlin)
           Translate resultant plan into frames
       • 2 d. Apply Learning Algorithm(s)
           Interpret plan as partially ordered network of
              actions such that primitive actions are
              algorithm calls
3. Evaluate Learning (not implemented)
```

Fig. 12. Introspective multistrategy learning algorithm.

actually make a deliberate test of the newly learned knowledge by trying to falsify the information. When either of these processes finish, the verification phase would output an evaluation of the quality of learning.

The most critical of the three phases above is the second phase (learning generation) that generates changes to the BK. Fig. 12 sketches the algorithm that implements these processes. The system records a trace of the reasoning used in the performance task in a sequence of trace meta-explanation structures. Each trace is inspected to detect a failure. When the system detects a failure, it invokes learning. During learning, the system constructs a learning strategy via the three process steps: blame assignment, deciding what to learn, and strategy construction. Subsequently, the system executes the learning strategy to perform the necessary knowledge repairs. These three steps are explained in more detail.

3.3.1. Blame assignment (step 2a, Fig. 12)

Take as input a trace of the mental and physical events that preceded a reasoning failure; produce as output an explanation of how and why the failure occurred, in terms of the causal factors responsible for the failure.

Blame assignment is a matter of determining what was responsible for a given failure. Thus, the function of blame assignment is to identify which causal factors could have led to the reasoning failure as determined from the output of the performance task and contained in the reasoning trace. That is, blame assignment is like troubleshooting; it is a mapping function from failure symptom to failure cause. The purpose is the same whether the troubleshooter is explaining a broken device or itself [1,93].

The input trace describes how results or conclusions were produced by specifying the prior causal chain (both of mental and physical states and events). The learner retrieves an abstract meta-explanation pattern, or IMXP, from memory and applies it to the trace in order to produce a specific description of why these conclusions were wrong or inappropriate. This instantiation specifies the causal links that would have been responsible for a correct conclusion, and enumerates the difference between the two chains and two conclusions (what was produced and what should have been produced). Finally, the learner outputs the instantiated explanation(s).

Step 2a of Fig. 12 outlines the control algorithm for blame assignment in an introspective multistrategy learner. The step is refined in Fig. 13 below. A characterization of the reasoning failure from the system's vocabulary of failure terms is used as an index to retrieve an ordered set of abstract IMXPs. The index is computed from the failure type returned by function *Detect-failure* and the vocabulary term of the trace node returned by the last stage of the reasoning process. Until success or the queue of IMXPs is empty, a candidate IMXP is bound and unified with the trace of the reasoning to produce a parameterized token. The PRE-XP-NODES (XP consequents) of the candidate are then checked to see if they are consistent with the current representation of the reasoning that produced an understanding the story (i.e., all are in the set of beliefs with respect to the FK). If they all can be verified then the Meta-XP applies to the situation. If any are rejected, then the explanation is rejected and a new candidate is tried.

Now the algorithm checks to see if any XP-ASSERTED-NODES (XP antecedents) are open. An open node is one for which neither the proposition nor its negation are believed, and thus the item does not have a representation in the model of the story. For each such node, Meta-AQUA tries to infer its belief by posing an introspective question concerning its existence. If the truth of any such node cannot be immediately determined, then the IMXP is indexed into memory, the blame assignment process is suspended, and the performance task is resumed. When future opportunities warrant, the system can resume the introspective process.

Finally, the system checks that no element of the set of XP-ASSERTED-NODES is contradicted by something in the story model or reasoning model within the FK. But if this is so, it tries to recursively explain the sub-anomaly. If the anomalous item cannot

```
Diagnose-failure (trace, failure-type)

Explanation-Queue ← Retrieve (trace, Compute-index (failure-type, trace))
LOOP
  If empty (Explanation-Queue) then
    return 'impasse
  else IMXP ← Front (Explanation-Queue)
  Unify (Trace-node(IMXP), trace)
  If [$\forall$X $\in$ Pre-XP-Nodes(IMXP) | X($\mathtt{in}_{FK}$)] then
    $\forall$ Y $\in$ XP-Asserted-Nodes(IMXP) | Y($\mathtt{out}_{FK}$) $\vee$ $\neg$Y($\mathtt{out}_{FK}$)
      Recursively pose the question "Is Y believable?"
      If cannot infer belief in Y then
        Index trace of reasoning in memory
        return 'suspend
    If [$\exists$ Z $\in$ XP-Asserted-Nodes(IMXP) | $\neg$Z($\mathtt{in}_{FK}$)] then
      explained-all-anomalies ← t
      $\forall$ Z $\in$ XP-Asserted-Nodes(IMXP) | $\neg$Z($\mathtt{in}_{FK}$)
        Recursively explain $\neg$Z
        If not explained then
          explained-all-anomalies ← $\emptyset$
      if explained-all-anomalies then
        return IMXP

    else return IMXP
```

Fig. 13. Reflective blame assignment.

be immediately explained, it rejects the explanation and returns for another one from the queue. If no anomalous nodes exist in the XP-ASSERTED-NODES or if all anomalies are explained, then the instantiated IMXP is accepted and returned. If no candidate explained the failure, then an impasse is signalled.

3.3.2. Deciding what to learn (step 2b, Fig. 12)

> *Take as input a causal explanation of how and why failure occurred; generate as output a set of learning goals which, if achieved, can reduce the likelihood of the failure repeating. Include with the output, both tentative goal dependencies and priority orderings on the goals.*

The previously instantiated Meta-XP assists in this process by specifying points in the reasoning trace most likely to be responsible for the failure. The Meta-XP also specifies the suggested type of learning goal to be spawned by this stage. A list of learning goals is included with each IMXP, and when the IMXP is bound to a TMXP, the instantiated goal list can simply be placed on a priority queue of current goals to be pursued by the next stage of the learning process. Because these goals are tentative, it may be necessary to retract, decompose, or otherwise adapt the learning goals dynamically during run-time. This stage of learning mediates between the case-based approach of blame assignment and the nonlinear planning approach of strategy construction. The learner includes with the output learning goals both tentative goal dependencies and priority orderings on the goals. The TMXP trace is passed as output also.

3.3.3. Learning-strategy construction (step 2c, Fig. 12)

Take as input a trace of how and why a failure occurred and a set of learning goals along with their dependencies; produce as output an ordered set of learning strategies to apply that will accomplish the goals along with updated dependencies on the set of goals.

The final learning-strategies are organized as plans to accomplish the learning goals. The plans are sequences of steps representing calls to standard learning algorithms. The plans are created by a Common LISP version of Tate's [96] Nonlin planner [33]. In order to use the nonlinear planner, the learning module translates the learning goals and the relevant context of the program environment to a predicate representation. In this form, Nonlin assembles a learning plan just as if it were creating a plan to stack a series of labeled blocks. The only difference is that the planner is given a set of learning operators that describe actions that modify the mental world (i.e., the BK) instead of the blocks world. These operators are written in the Task Formalism language defined by Tate (examples to follow below).

The learner instantiates the plan, translates it back into a frame representation, and, then executes the learning plans (in step 2d, Fig. 12). Nonlin generates a full order for the plan steps that achieves the conjunctive learning goals and avoids goal interactions. At the termination of the plan execution, control is returned to the performance system and story understanding is resumed.

4. Applying introspective multistrategy learning: Two extended examples

The following two examples illustrate the operation of the three phases of learning in an introspective multistrategy learner. They show the use of the representations described in Section 2 during execution of the algorithm described in Section 3. They also depict the kinds of failures encountered by the implementation during the trials to be reported in the evaluation of Section 5 and present operator representations of learning algorithms. The first example shows how Meta-AQUA learns when insufficient background knowledge exists to interpret an input. The second example shows how learning can function when the system forgets previous knowledge it learned.

4.1. Story one: A common contradiction

Given the drug-bust story of Fig. 14, the system attempts to understand each sentence by incorporating it into its current story representation, by explaining any anomalous or

Elvis was jonesing. Elvis took the lighter1 from the table2. He had the lighter1. The table2 didn’t have the lighter1. Police and dogs arrived. The phone1 was ringing. Mom picked up phone-receiver1. The phone1 wasn’t ringing. She had phone-receiver1. She let go of phone-receiver1. She didn’t have phone-receiver1. Officer1 went to outside. She [Mom] pushed light-switch1. The light1 was on. The cat1 pushed the vase2 to the floor1. The vase2 was broken. The police-dog1 went to outside. He [Officer1] pushed door-bell-switch1. The door-bell1 was ringing. He didn’t push door-bell-switch1. The door-bell1 wasn’t ringing. He went to the kitchen. The police-dog1 went to the kitchen. The police-dog1 went to the vase3. (S25) *The police-dog1 sniffed the vase3*. (S26) *The police-dog1 barked at the vase3*. The police-dog1 was barking. He [Officer1] went to the vase3. He took the ganja1 from the vase3. He had the ganja1. The vase3 didn’t have the ganja1. (S32) *He arrested Elvis*. He controlled Elvis. He went to outside. Elvis went to outside. The police-dog1 went to outside. (S37) *If the police-dog1 detects the ganja1 then the police-dog1 will bark at the vase3.* He [Elvis] was still jonesing.

— The End —

Fig. 14. Tale-Spin story TS2.

interesting features of the story, and by learning from any reasoning failures. Numerous incorrect inferences can be made from this story, depending on the knowledge of the reader. Meta-AQUA’s background knowledge includes general facts about dogs and sniffing, including the fact that dogs bark when threatened, but it has no knowledge of police dogs. It also knows cases of gun smuggling, but has never seen drug interdiction.

The story is processed smoothly using scriptual knowledge that relate actors, their goals and the kinds of actions that achieve goals. For example, sentence S25 produces no inferences other than that sniffing is a normal event in the life of a dog. However, S26 produces an anomaly because the system’s definition of “bark” specifies that the object of a bark must be animate. The program (incorrectly) believes that dogs bark only when threatened by animate objects. Since vase is inanimate, there is a conflict. This anomaly causes Meta-AQUA to ask itself why the dog barked at an inanimate object. Given a prior explanation about dogs barking when threatened by persons, it hypothesizes that the vase somehow threatened the dog. It suspends the question, however, after it no longer can proceed due to the lack of additional information. S32 posits an arrest scene that reminds Meta-AQUA of an incident in which weapons were smuggled by terrorists; however, the sentence generates no new inferences concerning the previous anomaly. Finally, S37 causes the original question generated by S26, “Why did the dog bark at the luggage?” to be retrieved. Instead of revealing the anticipated threatening situation, however, S37 offers another hypothesis. The detection of drugs in the vase caused the dog to bark.

At this point, the system has detected an explanation failure, and so it suspends the performance task. Until now, all processing was first-order reasoning about the story using first-order knowledge about the domain of criminal activities. Learning involves second-order reasoning about the prior, faulty story understanding effort using second-order knowledge about failures and about the processes in the first-order task.

4.1.1. Blame assignment: Explaining reasoning failure

The system characterizes the reasoning error as a contradiction caused by the incorrect retrieval of a known explanation (“dogs bark when threatened by objects”, erroneously assumed to be applicable), and a missing explanation (“the dog barked because it detected marijuana”, the correct explanation in this case). During blame assignment, Meta-AQUA uses this characterization as an index to retrieve an abstract case (Meta-XP) that is applied to a trace of the reasoning that produced the failure. Fig. 15 shows the instantiated result in an explanation of its reasoning error. This composite meta-explanation consists of three parts: a Novel-Situation centered around `Retrieval Failure`, an Erroneous-Association centered around `Expectation Failure` and an Incorrect-Domain-Knowledge centered around `Incorporation Failure`.

The abstract IMXP from which this instantiation originates, IMXP-NOVEL-SITUATION-ALTERNATIVE-REFUTED, captures a common pattern of failure in systems that are learning new concepts. When a concept is being learned, it may be overly specialized. Slight variation on the concept will cause the system to try to explain it, but without experience with the concept, the system may generate an inappropriate explanation. The proper explanation may not be known because the situation is novel. Notice that IMXP-NOVEL-SITUATION-ALTERNATIVE-REFUTED applies equally to both the failed explanation for why Lynn hit the ball and for the current failed explanation for why the dog barks at the vase.

As seen in Fig. 15, the vertical chain of processes starting with the node labeled “Pose Question” represents part of a TMXP. Following causal links 4 through 7, the trace represents the sequence in which the question “Why did the dog bark at the vase?” enables the explanation that the dog barked because it was threatened and therefore results in the search for corroboration. This trace records the decisions preceding the detection of the explanation failure (i.e., that the dog was actually barking because it detected the contraband, not out of defensive instincts). The IMXP structure formally explains the node labeled `Expectation Failure`, although in general, it gives the causal chain of events for much of the reasoning associated with all parts of the error.[19] To check whether or not this explanation applies to the failure, Meta-AQUA checks the truth values of nodes A_2, E, and the EXPLAINS node (i.e., the PRE-XP-NODES of the IMXP). Because these already exist as known entities in the representations (i.e., $\mathtt{in}_{\mathtt{FK}}$), the XP is accepted.

Although the information in Fig. 15 appears to be complex, the IMXP simplifies the amount of detail the reasoner must consider during blame assignment by abstracting away much of this information. To show what the system actually considers, Fig. 16 represents an overlay that corresponds to Fig. 15 (mentally align the shaded nodes, such as the `Incorporation Failure` marked IF, between the two figures to see the simplification Fig. 16 provides).

4.1.2. Deciding what to learn: Spawning learning goals

Given a reasoning failure, the learning task is to adjust a system’s knowledge so that such reasoning failures will not recur in similar situations. To perform the adjustment, the

[19] Note that the node labeled “EXPLAINS” in Fig. 15 is the EXPLAINS node for the XP labeled A_2, not for the IMXP itself.

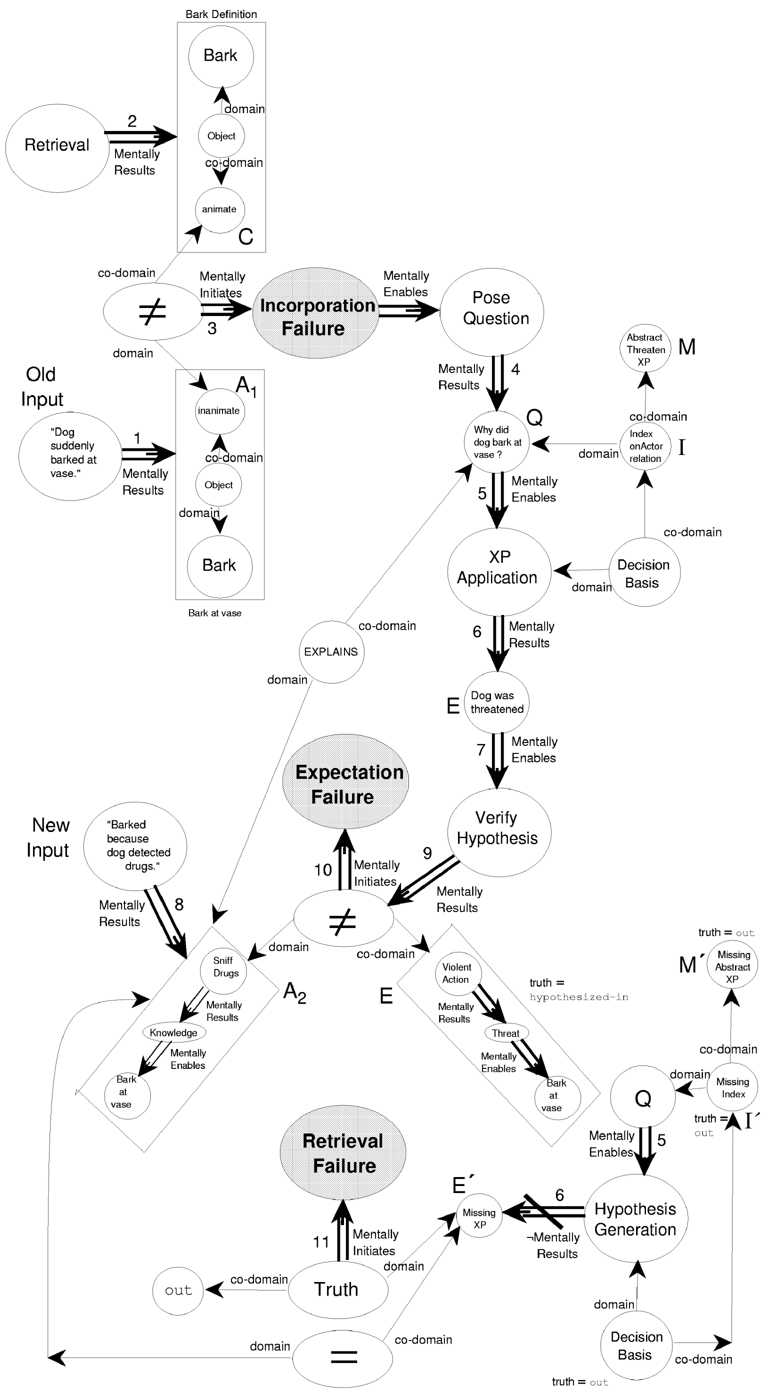

Fig. 15. Instantiated IMXP for mis-explanation.

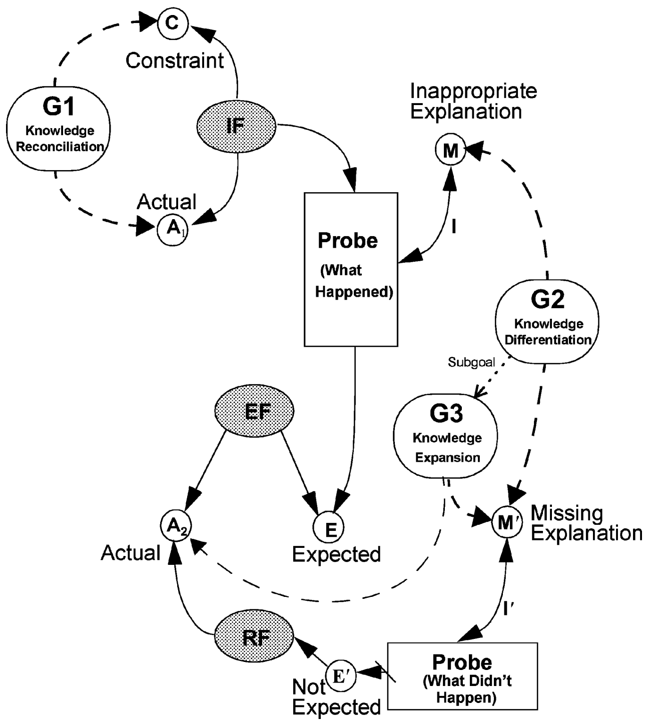

Fig. 16. Abstracted IMXP with learning goals.

learner constructs a learning strategy that is designed to satisfy specific learning goals. These goals are created in a decision process that circumscribes what needs to be learned. The decision is made in response to the explanation of failure generated during reflective blame assignment.

Faced with the structure of the reasoning error produced by the blame assignment phase, the learner determines the learning goals for the system. First, since the seemingly anomalous input (marked "Old Input" in Fig. 15 or A_1 in Fig. 16) has been incorporated into the story and later reinforced by the coherence of the story structure, and since no contradictions occurred as a result of this inference, the learner posts a knowledge reconciliation goal (G1 in Fig. 16). The goal is to adjust the background knowledge so that neither dogs barking at animate objects nor dogs barking at inanimate objects will be considered anomalous by the understander. This learning goal is appropriate because even though one item is an instantiated token (a particular dog barked at a specific inanimate object), while the other is a type definition (concept specifying that dogs generally bark at animate objects), they are similar enough to each other to be reconcilable.

Secondly, given that an expectation failure triggered the learning, and (from the blame assignment phase) given that the failure resulted from the interaction of misindexed knowledge and a novel situation, Meta-AQUA posts a goal to differentiate between the two explanations for why the dog barked (nodes M and M′ in Fig. 16). Since the conflicting explanations are significantly different (for example, they do not share the same predicate, i.e., detect versus threaten), a knowledge differentiation goal is licensed, rather than a goal to reconcile the two types of explanations. The differentiation goal is achieved if the system can retrieve proper explanations given various situations. The original misunderstanding of the story occurred, not because the explanation that dogs bark when threatened is incorrect in general, but rather because the system did not know the proper conditions under which this explanation applies.

In addition to posting these two learning goals, Meta-AQUA places a tentative ordering on their execution. With no other specific knowledge concerning their respective relations, a good default heuristic is to order them by the temporal sequence of the failures involved in the original reasoning trace. This heuristic is useful because if it is determined that the first failure was not an error but either was a misunderstanding or was caused by faulty input, then the reasoning that followed from the first failure (or other assumptions depending on the nature of the first failure that led to the second) may have contributed to the cause of the second. Thus, learning acquired about the first failure may show that the subsequent reasoning was irrelevant, or it may yield more information to be used on the second goal. Hence, the stage that decides what to learn outputs the knowledge reconciliation goal with priority over the knowledge differentiation goal.

4.1.3. Learning-strategy construction: Learning as a planning task

Blame assignment during the drug-bust example retrieved an IMXP that explained the faulty explanation for why the dog barked at the vase. Instead of barking because it was threatened by the vase, the dog barked because it detected contraband in the vase. The system then spawned two learning goals: G1, a knowledge reconciliation goal, and G2, a knowledge differentiation goal (see Fig. 16). The learner thus must reconcile the input (previously believed to be faulty) with its conceptual definition of dog-barking, and it must differentiate the two explanations so that neither is confused for the other.

Planning for a knowledge differentiation goal

Consider the knowledge differentiation goal, G2, of Fig. 16. It seeks to differentiate between the expected explanation that the dog barked because it was threatened and the actual explanation that the dog barked because it detected contraband. This goal can be achieved by reindexing the memory locations for the two explanations so that they will be retrieved when appropriate. However, because the system has no prior experience with the actual explanation, A_2, (and thus the system neither foresaw nor considered the explanation), the learner posts a subgoal to expand the instantiated explanation (i.e., the knowledge expansion goal G3) to produce the missing explanation pattern, M'.

The schemas that produce the subgoal sequencing of events are defined in Fig. 17. The `use-when` conditions on the *index-xp* operator guarantees that this schema is chosen only if the variable?*x* is both an XP and is a token (instance) rather than an type (abstract explanation). The *gen-op* operator does not actually decide which algorithm to perform the generalization; the *do-generalize* action schema does. Explanation-based generalization (EBG) [26,60] can be selected as an appropriate learning algorithm for this task.

A more difficult problem is to differentiate the applicability conditions for the two abstract explanations (M', the one produced by generalizing the detection explanation, A_2,

```
;;; To index in memory some XP that is a token,
;;; first spawn a subgoal to expand the concept
;;; into a type, then perform the (non-primitive)
;;; action to index the XP.
;;;
(opschema index-xp-op
    :todo (indexed ?x)
    :expansion (
             (step1 :goal (knowledge-expansion ?x))
             (step2 :action (index-item ?x))
           )
    :conditions (
             (:use-when (isa xp ?x) :at step1)
             (:use-when (isa token ?x) :at step1))
    :orderings ( (step1 -> step2) )
    :variables (?x)
)
```

```
;;;
;;;
;;; To achieve a knowledge expansion goal,
;;; perform some kind of generalization.
;;;
(opschema gen-op
       :todo        (knowledge-expansion ?x)
       :expansion   (
                      (step1 :action (do-generalize ?x))
                     )
       :variables   (?x)
       )
```

Fig. 17. Schema definitions to index an XP.

and M, the original XP that produced the initial threaten explanation, E) by modifying the indexes (I′ and I) with which the system retrieves those explanations. If the two problems of erroneous association and novel situation were to be treated independently, rather than as a problem of interaction, then an indexing algorithm would not be able to ensure that the two explanations would remain distinct in the future. That is, if the learner simply detects a novel situation and automatically generalizes it, then indexes it by the salient or causal features in the explanation, and if the learner independently detects an erroneous retrieval, and re-indexes it so that the same context will not retrieve it in the future, then there is no guarantee that the resultant indexes will be mutually exclusive. Instead, the system must re-index M *with respect to* M′, not simply with respect to the condition with which M was retrieved.

The problems to be solved, then, are determining the difference between M and M′, and, in the light of such differences, computing the minimal specialization of the index of M and the maximally general index of M′ so they will be retrieved separately in the future. In the case of the drug-bust story, HC1, the problem is somewhat simplified. The difference is that retrieval based on the actor relation of barking actions (dogs) is too general. The threat explanation applies when dogs bark at animate objects. The detection explanation is appropriate when dogs bark at containers.

Fig. 18, “Mutual-indexing schemas”, shows learning-operator definitions for the indexing strategy that manages mutual indexing between two concepts. First, the operator schema determines that both concepts must be independently indexed before they are indexed with respect to each other. The first two conditions (`precond`) assure that if they are not already indexed then subgoaling will be forced at steps one and two. The action schema has filter conditions (`use-when`) that enable the schema only when the concepts are both indexed XPs.

Additionally, the effects list of *do-mutual-xp-indexing* includes the deletion of the predicate *clear2change*. By doing so, any other operator that requires the explained action to be “clear to change” will be placed before the mutual indexing step of the final learning plan. That is, a linearization will be performed on external goals to reorder any other schema that has the predicate as an unsupervised condition (`unsuperv`). This ordering constraint is similar to the interaction between the goal to stack A on B and the goal to stack B on C when all three blocks lie on the table. In the blocks world case, the *puton*(X, Y)

```
;;;
;;; To index 2 items with respect to each other,
;;; make sure both are indexed independently,
;;; then index them jointly.
;;;
(opschema mutual-index-op
       :todo   (index-wrt-item ?x ?y)
       :expansion (
              (step1 :goal (indexed ?x))
              (step2 :goal (indexed ?y))
              (step3 :action
                     (index-dual-items ?x ?y)))
       :orderings(
              (step1 -> step3)
              (step2 -> step3))
       :conditions (
              (:precond (indexed ?x)
               :at step3 :from step1)
              (:precond (indexed ?y)
               :at step3 :from step2)
              (:use-when (not (equal ?x ?y))
                     :at step1))
       :effects   ()
       :variables (?x ?y))
```

```
;;; Action of indexing 2 explanations jointly requires
;;; that any changes to definition of the parent type
;;; of the domains of their explains-node (that is, the
;;; explained-action) be performed before indexing is.
;;;
(actschema do-mutual-xp-indexing
       :todo   (index-dual-items ?x ?y)
       :expansion ( (step1 :primitive
                     (perform-mutual-indexing ?x ?y)))
       :conditions (
              (:use-when (indexed ?x) :at step1)
              (:use-when (indexed ?y) :at step1)
              (:use-when (isa xp ?x) :at step1)
              (:use-only-for-query
                     (explains ?explains-node ?x)
               :at step1)
              (:use-only-for-query
                     (domain ?explains-node
                      ?explained-action)
               :at step1)
                            )
       :effects  (
              (step3 :assert (indexed-wrt ?x ?y))
              (step3 :assert (indexed-wrt ?y ?x))
              (step3 :delete
                     (clear2change ?explained-action)))
       :variables
            (?x ?y ?explains-node ?explained-action))
```

Fig. 18. Mutual-indexing schemas.

schema requires both blocks X and Y to be "clear" and then deletes the clearness condition of the destination block Y. Thus if A is put on B, block B is not clear to put on C. Instead, B must first be put on C. In effect, the *clear2change* state requires that all attributes of its argument be stable before the schema operates on the argument; no other operator can change an attribute in order for the changes performed by indexing to be unaffected.

Planning for a knowledge reconciliation goal

In Fig. 16, the remaining learning goal, G1, represents a knowledge reconciliation goal. The goal is to reconcile the fact that the conceptual definition of dog-barking is limited to animate objects with the fact that a particular dog barked at a vase. This goal can be thought of as a simple request for similarity-based learning or inductive learning (e.g., UNIMEM's algorithm in [49], or abstraction transmutation as in [57]). The system is simply adding an additional positive example to the instances seen. An incremental algorithm is required because this instance has been discovered after an initial concept has been established some time in the past.

An interesting interaction can occur, however, if the system waits for the result of the EBG algorithm required by the knowledge expansion subgoal G3. The algorithm will generalize the explanation (that this particular dog barked at a particular vase because it detected marijuana) to a broader explanation (that dogs in general may bark at any container when they detect contraband). Thus, the example provided to the inductive algorithm can be more widely interpreted, perhaps allowing its inductive bias to generalize the constraint, C, on the object of dog-barking to `physical-object` (the exhaustive case of `animate-object` and `inanimate-object`), whereas a single instance of a particular breed of dog barking at a specific vase, A_1, may limit the inductive inference if no additional domain knowledge is available.

Unfortunately, however, because the EBG algorithm uses the representation of the dog-bark definition, and the inductive algorithm changes this definition, the induction must

```
;; Perform an abstraction transmutation on relation r1 given relation r2.
;; The function raises the co-domain of r1 to the shared parent type of r1 and r2.
;; In the example, r1  = bark-def
;;;                     -> (object (domain dog-bark)(co-domain animate-obj))
;;;                r2  = bark-example
;;;                     -> (object (domain dog-bark)(co-domain inanimate-obj))
;;;
(actschema do-abstraction-change
       :todo  (abstracted ?r1 ?r2)
       :expansion ( (step1 :primitive (perform-abstraction ?r1 ?r2)))
       :conditions (
              (:use-when (isa relation ?r1) :at step1)
              (:use-when (isa relation ?r2) :at step1)
              (:use-when (relation ?r1 ?r1-type) :at step1)
              (:use-when (relation ?r2 ?r2-type) :at step1)
              (:use-only-for-query (domain ?r1 ?r1-domain) :at step1)
              (:use-only-for-query (co-domain ?r1 ?c) :at step1)
              (:use-only-for-query (co-domain ?r2 ?a) :at step1)
              (:use-only-for-query (parent-of ?c ?c-parent) :at step1)
              (:use-only-for-query (parent-of ?a ?a-parent) :at step1)
              (:use-when (equal ?r1-type ?r2-type) :at step1)
              (:use-when (equal ?c-parent ?a-parent) :at step1)
              (:unsuperv (clear2change ?r1-domain)  :at step1))
       :effects  (
              (step1 :assert (co-domain ?r1 ?c-parent))
              (step1 :delete (co-domain ?r1 ?c)))
       :variables (?r1 ?r2 ?r1-type ?r2-type ?r1-domain ?c ?a ?c-parent ?a-parent))
```

Fig. 19. Abstraction schema.

occur first. Thus, the system cannot take advantage of the opportunity cited in the previous paragraph. One important implication of this point is that in systems which plan to learn, if the reasoner does not anticipate this second interaction (thus placing EBG before the induction), the system must be able to perform dynamic backtracking on its decisions.

Like a nonlinear planner in the blocks world, the learning system must detect any dependency relationships so that goal violations can be avoided. For example, when the definition of dog-barking is modified by abstracting the constraint on the objects at which dogs bark from `animate-object` to `physical-object`, any indexing based on the modified attribute must occur after this modification, rather than before it, to avoid indexing with obsolete conceptual knowledge. Note that the action schema of *do-abstraction-change* in Fig. 19 has an unsupervised condition requiring the `?r1-domain` variable to be “*clear2change*”. As discussed in section, this will cause the abstraction to be performed prior to any step that deletes the state (e.g., the *do-mutual-xp-indexing* schema of Fig. 18). Therefore, if both schemas are being instantiated, the nonlinear planning module of Meta-AQUA will automatically order the abstraction before the indexing. Because the generalize schema also deletes the *clear2change* state, generalization of the detection explanation is prevented from occurring before the abstraction.

4.1.4. Strategy execution: Performing the learning and the aftermath

After a learning plan is constructed, a very simple process can execute the plan. All primitive steps in the plan are calls to learning algorithms from the toolbox. Because the plans are partially ordered, not all steps will have a linear order enforced. Therefore, some steps may be executed in parallel. In the drug-bust example of story TS2, however, the

Symptoms:

- Contradiction between input and background knowledge
- Contradiction between expected explanation and actual explanation

Faults:

- Incorrect domain knowledge
- Novel situation
- Erroneous association

Learning Goals:

- Reconcile input with conceptual definition
- Differentiate two explanations
- Acquire new XP (subgoal)

Learning Plan:

- Abstraction on concept of bark
- Generalization on bark explanation
- Index new explanation
- Mutually re-index two explanations

Fig. 20. Learning from Explanation Failure.

final learning plan Meta-AQUA constructs is fully ordered. The output from Nonlin is a plan whose resultant steps are:

(1) perform an abstraction transmutation on the concept of dog barking (realizing that dogs bark at containers);
(2) perform EBG on the new explanation (producing a generalized version);
(3) index the generalized XP in isolation; and finally,
(4) use the new concept definition to mutually differentiate and index the two generalized explanations of why dogs bark.

This plan is translated back into a frame representation and executed in the order specified. Fig. 20 lists the major state transitions that the three learning processes produce.

After the learning, control is returned to the story understanding module. The system continues with the story until completion. In subsequent stories (or even within the same story), similar types of failures should not repeat when the learning is successful. For example, Meta-AQUA is given a story in which a police officer and a canine that enter a suspect's house, the dog barks at the refrigerator, and the suspect is arrested for possession of some discovered marijuana. The new story causes no anomaly when the dog barks at the inanimate container. Indeed, Meta-AQUA expects some type of contraband to be found in the refrigerator after it reads that the dog barked, but before it is told of the contraband's existence. Thus, the learning accomplished in the previous story improves both understanding of and predictions for subsequent stories.

4.2. Story number two: A baffling situation (impasse)

Even when a system learns new concepts or changes old ones, the successful use of this information may not always occur if the BK is not organized to promote the retrieval of it when needed. A retrieval impasse can occur because, given the cues that compose a probe into memory, the existing explanation can lack the proper index with which to traverse the BK and thus to find the item. In effect, Meta-AQUA can forget learned explanations and can expect instead to acquire this knowledge anew. When Meta-AQUA is given the correct old explanation, the system must be reminded of the explanation that already exists in memory. Therefore, rather than pursuing a goal to acquire and expand the knowledge, it must be able to switch to a goal of reorganizing the knowledge. That is, learning goals are not static; even a system that uses the introspective method of deciding what to learn must be prepared to change its learning goals dynamically.

In story TS2, the police dog barked at a vase in Elvis' home. Meta-AQUA considered the event to be anomalous because the system believes that dogs bark only at animate objects. As was seen in the previous section, the program eventually learned that dogs can bark at any physical object, including inanimate ones. It also learned the new explanation that "dogs bark when detecting contraband". So after processing TS2, Meta-AQUA's memory contains knowledge of two explanations for why dogs bark: an explanation for dogs that bark because they are threatened (indexed by `dog-barks-at-animate-object`) as well as an explanation for dogs that bark because they detect contraband (indexed by `dog-barks-at-container`).

Tale-Spin then generates the subsequent story TS3 (see Fig. 21) and outputs it to Meta-AQUA. In this story, Elvis and Lynn are about to play with the ball when the police arrive at the house with a canine unit. The dog immediately goes to a throw-rug and sniffs at the object (S23). When the dog barks (S24) the officer pulls back the rug to find Elvis' stash of marijuana. Consequently, the officer arrests Elvis (S30) and takes him away (S32, S33). The story then informs the reader that the dog barked because it detected the contraband (S35). Because Elvis looses his freedom due to the arrest, he can no longer play ball with Lynn, and so he remains bored (the original problem that motivated the story).

Immediately after the dog barks at the carpet, Meta-AQUA generates a question to explain why the dog barked. The reason for this decision is that the system has recently learned about dogs and barking, so it is interested in any subsequent information that may be related. However, because the dog is barking at a rug and such an object is not a container, it does not retrieve the newly learned detection explanation. The dog also is not barking at an animate object, so the old threaten explanation is not retrieved. Instead, it can generate no explanation to explain the interesting story concept.

Reviewing the reasoning trace that preceded the conclusion, Meta-AQUA characterizes itself as "baffled" (impasse during memory retrieval). The system retrieves an IMXP based on this characterization, which helps it explain its own reasoning failure. The structure is unified with the representation of the original reasoning (stored in a TMXP) which produces the instantiation partially shown in Fig. 22. The knowledge structure shows that memory retrieval produced no explanation in response to the system's question. Instead, a later input produced the answer.

Elvis was bored. Elvis asked Lynn, “Would you push the ball1 to me away from you?” Police-and-dogs arrived. Officer1 went to outside. The police-dog1 went to outside. He pushed door-bell-switch1. The door-bell1 was ringing. He didn’t push door-bell-switch1. The door-bell1 wasn’t ringing. The phone1 was ringing. Mom picked up phone-receiver1. The phone1 wasn’t ringing. She had phone-receiver1. She let go of phone-receiver1. She didn’t have phone-receiver1. The cat1 pushed the vase2 to the floor1. The vase2 was broken. She pushed light-switch1. The light1 was on. He went to the kitchen. The police-dog1 went to the kitchen. The police-dog1 went to the rug1. (S23) *The police-dog1 sniffed the rug1.* (S24) *The police-dog1 barked at the rug1.* The police-dog1 was barking. He went to the rug1. He took the ganja1 from the rug1. He had the ganja1. The rug1 didn’t have the ganja1. (S30) *He arrested Elvis.* He controlled Elvis. (S32) *He went to outside.* (S33) *Elvis went to outside.* The police-dog1 went to outside. (S35) *If the police-dog1 detects the ganja1 then the police-dog1 will bark at the rug1.* He [Elvis] was still bored.

— The End —

Fig. 21. Tale-Spin story TS3.

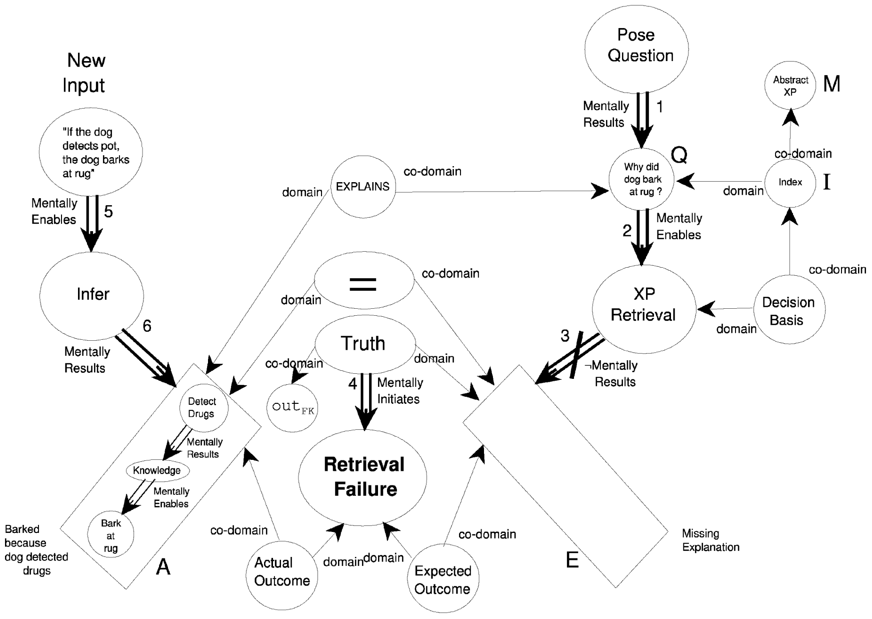

Fig. 22. Instantiated forgotten detection explanation.

The IMXP suggests that a knowledge expansion goal be spawned to generalize the input explanation. This suggestion comes from the *potential-learning-goal* slot of the IMXP (see Fig. 23). Conditions attached to the knowledge expansion goal allow it to be posted if the node A was either acquired from the story or inferred, but not if it was retrieved from

```
(define-frame IMXP-BAFFLED-AND-RESOLVED
   (isa           (composite-introspective-meta-xp))              ;; IMXP Class
   (failure-cause (novel-situation.0 missing-assoc.0))            ;; Which one we do not know
   (q             (relation(explanations (=a))))                  ;; Baffling question
   (a             (xp    (explains =q)))                          ;; Actual explanation
   (e             (xp    (explains =q)))                          ;; Missing expectation
   (i             (index (domain =q) (co-domain =m)))             ;; Index used to retrieve E
   (m             (xp))                                           ;; Forgotten xp.
   (truth-value   (truth (domain =e)                              ;; E not in set of beliefs wrt FK
                         (co-domain out-fk.0)))
   (equals        (equal-relation(domain =a)                      ;; Actual should have been
                         (co-domain =e)))                         ;; equal to what was expected
   (rf            (retrieval-failure(initiates- =truth-value)     ;; The memory failure
                         (expected-outcome =e)                    ;; explained by the IMXP
                         (actual-outcome =a)))
   (new-input     (entity))                                       ;; Story input
   (later-process (cognize))                                      ;; Inference in this case
   (rc            (trace-meta-xp                                  ;; Reasoning chain
                         (identification =q-id)
                         (generation =hypo-gen)
                         (link3 =link2)
                         (link4 (mentally-results (truth out-fk.0)))))
   (q-id          (d-c-node                                       ;; Question identification
                         (strategy-choice questioning.0)
                         (strategy-execution pose-question.0)
                         (side-effect (considerations =con
                                (prime-state =k-goal)))
                         (link4 =link1)))
   (k-goal        (knowledge-acquisition-goal                     ;; Knowledge goal to answer
                         (goal-object                             ;; the question
                         (generate (co-domain =q)))))
   (hypo-gen      (d-c-node                                       ;; Hypothesis generation
                         (strategy-decision =h-decision)
                         (main-result (outcome =o (members (=link4))))
                         (link4 (mentally-results (co-domain =o)
                         (truth out-fk.0)))))
   (h-decision    (decision-process                               ;; XP retrieval in this case
                         (basis-of-decision =h-decision-basis)))
   (h-decision-basis
                  (basis (knowledge                               ;; Existence of I is the basis
                                (collection                       ;; to use case-based explanation
                                (members ((knowledge-state
                                            (co-domain =i)
                                            (believed-item =i)))))))
   (links         (=link1 =link2 =link3 =link4 =link5 =link6))    ;; Links are in temporal order
   (link1         (mentally-results (domain pose-question.0)
                            (co-domain (outcome (members (=q))))))
   (link2          (mentally-enables  (domain =con)
                            (co-domain =hypo-gen)))
   (link3         (mentally-results  (domain =rc)                 ;; and all correspond to the
                            (co-domain =e)))                      ;; numbered links in Fig. 22.
   (link4          (mentally-initiates (domain =truth-value)
                            (co-domain =rf)))
   (link5         (mentally-enables  (domain =new-input)
                            (co-domain =later-process)))
   (link6         (mentally-results  (domain =later-process)
                            (co-domain =a)))
   (explains      =rf)                                            ;; What the IMXP explains.
   (pre-xp-nodes (=a =e =rf))                                     ;; XP consequents.
   (internal-nodes (=q =hypo-gen =later-process =i))              ;; Neither sink nor source nodes
   (xp-asserted-nodes (=q-id =m =new-input))                      ;; XP antecedents.
   (potential-faults (=a =i))                                     ;; Nodes for blame-assignment
   (potential-learning-goals                                      ;; Corresponding learning goals
                  ((knowledge-expansion-goal
                         (goal-object =a)                         ;; Expand the new explanation
                         (subgoals =krg)
                         (priority (integer-value =pr))
                         (backptr (plan))
                         (conditions ((inferred.0 acquired.0)))
                   (knowledge-reorganization-goal =krg
                         (goal-object =i)                         ;; Reorganize memory to hold it
                         (priority (integer-value (less-than =pr))))))))))
```

Fig. 23. IMXP frame definition for forgetting.

memory. A knowledge organization goal is also spawned in order to index the generalized explanation in memory. These goals can be achieved by performing explanation-based generalization (EBG) on the new explanation (node A) and then indexing the explanation by the context in which the system encountered the explanation.

The system cannot determine *a priori* whether an abstract XP (node M) actually exists in memory but could not be recalled (thus, the failure cause is a missing association, I), or whether the system lacks the knowledge to produce the explanation (thus, the cause is that the situation is novel, i.e., M is missing). The system thus poses a question about its own IMXP (cf. [64]), "Does M exist in memory?" If M is missing, I is also missing; thus, the right question to ask is whether M exists, not I. [20]

The answer to the introspective question is obtained by performing EBG and then watching for a similar explanation in memory when it stores the new explanation via the indexing algorithm. The system can detect the presence of similar memories by maintaining a list of pointers to memory items for each conceptual type. At storage time, Meta-AQUA traverses the list, checking each to see if it can unify the new memory with any of the older ones. [21] Meta-AQUA thus finds the explanation produced by the previous story.

Merging the two explanations produces a better explanation: Dogs may bark at objects that hide contraband, not just at containers that hold contraband. The algorithm that indexes the generalization searches for the common ancestor of the object slots of both explanations (i.e., objects that contain other objects and objects that cover other objects). This common ancestor is the type `hiding-place`. Thus, so that these types of explanations will not be forgotten again, the system indexes the explanation by "dogs that bark at potential hiding places" and places a pointer to the merged explanation on the memory list for the symbol `causal-relation`.

As a result of its learning, Meta-AQUA reduces the number of anomalies generated in similar stories, and it improves its predictive explanations. Most importantly, however, this story illustrates the fact that learning goals are not static, but rather, that they are subject to dynamic re-evaluation, even when the planner that creates a learning plan knows about interactions. Some facets that bear on the pursuit of learning goals cannot always be anticipated in advance. In the example, the system decides that it should acquire a new piece of knowledge, but instead it discovers that it already has the knowledge in memory. Instead of achieving a knowledge expansion goal to generalize and store the supposed new explanation, it rediscovers the old one and changes the learning goal to a knowledge organization goal. As such, it represents a simple example of an autonomous goal transformation [23].

[20] Note that it cannot be the case that I is erroneous. If it were true, then some explanation would have been retrieved, although it may have been inappropriate. Cox [13] enumerates a number of constraints on blame assignment.

[21] This mechanism simulates a memory such as that of DMAP [51,52], whereby memory items map to areas that contain similar memories. Although Meta-AQUA's mechanism is only a crude approximation to such architectures, the emphasis of our theory is on the reasoning about memory (or other reasoning processes), rather than on a representation of the memory architecture *per se*. A more realistic mechanism would be for Meta-AQUA to use the generalized XP as a probe to memory to see if it is now reminded of the old XP. The current method suffers from the fact that it always finds the old XP at an unacceptable search cost.

5. Computational evaluation

This section presents the results of computational studies performed with Meta-AQUA to test the assertion that the second phase of learning (deciding what to learn) is *necessary* for effective learning. Few computational systems other than Meta-AQUA include an explicit calculation of a goal to learn and then use that goal to influence learning (e.g., the theory of intentional learning embodied in CELIA [81] entails implicit learning goals). Converging with the hand-coded examples from previous research that favor this position (e.g., [10,21]), this paper presents quantitative evidence that supports the utility of this stage (see also [12]). More specifically, we assert that the rate of improvement in story understanding with learning goals exceeds that of story understanding without learning goals holding all other factors constant. Removing the learning goals eliminates part of the system's mechanism responsible for introspection. The intention of this manipulation is to show different empirical learning curves with and without introspection as a function of the number of inputs.

First we describe the space of reasoning failures exhibited by the system during the evaluation. Next, the second subsection defines the independent and dependent variables being measured. Finally, the third subsection reports the data and their analysis.

5.1. The implemented space of explanation failures

As currently implemented, the blame assignment phase of learning does not consider all of the failure causes enumerated in Table 2, "Taxonomy of causes of reasoning failure", although the implementation does consider many more of these causes than do most AI systems. At the present time, the system concentrates on errors that arise from missing and flawed domain information and the indexing of such information in the BK, that is, the "Knowledge states" columns of Table 2. Yet given this limitation, the combinations of failure encountered are many (see Fig. 24) and the resultant learning can be nontrivial.

Fig. 24 graphically illustrates the space of failure causes that blame assignment considers in the experimental study presented here. The large shaded portion of the figure represents Meta-AQUA's performance system when no failures are detected. The program first accepts a given input. If the input is anomalous, the system explains it, otherwise it checks to see if the input is in any other way interesting. If it is interesting, the system explains it; otherwise, it skims the input and accepts another. Outside of the shaded area represents the space of failures.

When Meta-AQUA generates an explanation for an anomaly in the input story, the explanation may be incorrect. Alternatively, it may reach an impasse when trying to explain and thus not be able to generate an explanation at all. In the first case the explanation may be wrong, but the right explanation was in memory all along. If it cannot explain an anomaly, the explanation may have existed, but could not be found. All of these cases can occur both when the input is anomalous (left hand side of Fig. 24) and when simply interesting (right hand side of the figure). An additional case occurs when Meta-AQUA explains an anomalous input correctly. It can then learn what is wrong with its knowledge that leads it to believe an anomaly exists. One may object that because these all map to a single fault, a decision tree could be built rather than going through the introspective

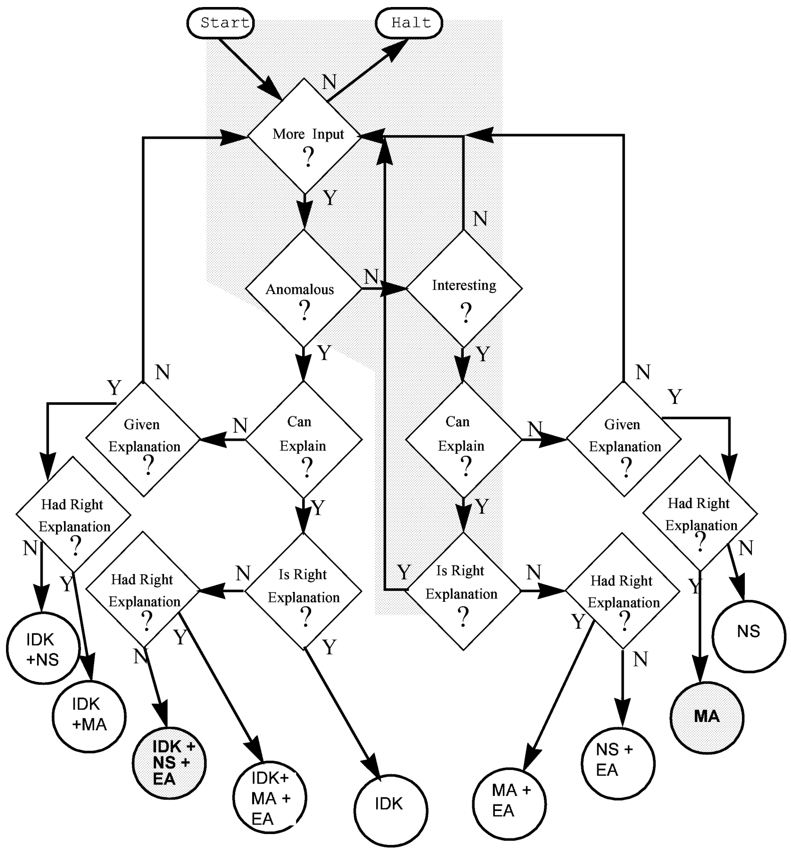

Fig. 24. Implemented space of reasoning faults. Circles list failure combinations that occur given a situation: IDK = incorrect domain knowledge; NS = novel situation; MA = missing association; EA = erroneous association. The left-most shaded circle matches the fault in story TS2 and the right-most matches TS3 (although, in TS3, Meta-AQUA first thought the fault was NS).

process. However, this figure represents the conditions available only in hindsight or through the auspices of an oracle; it is a virtual flow-chart, not an actual flow of control in the program. Meta-AQUA must go through the blame-assignment process in order to determine the actual situation that applies to a given set of circumstances. As illustrated in Story number two of Section 4.2, a system cannot determine *a priori* that it has the right explanation in memory but failed to find it (i.e., forgotten the explanation due to a missing association). A set of if-then statements will not suffice to perform blame assignment.

5.2. Independent and dependent variables

Introspective learning is a computational process with the decomposition as shown in the upper portion of Fig. 25. *Fully introspective multistrategy learning* consists of examining one's own reasoning to explain where the reasoning fails. It consists further of knowing

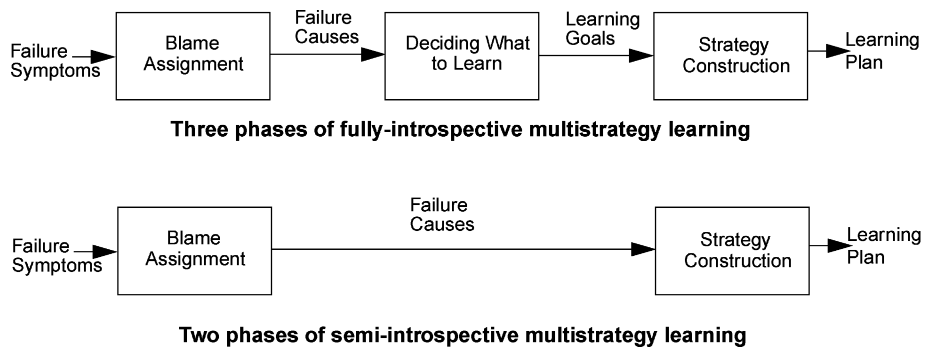

Fig. 25. Learning goal ablation.

enough about the self and one's own knowledge that the reasoner can explicitly decide what needs to be learned. Removing the goals from the introspective process leaves a more reflexive activity we call *semi-introspective multistrategy learning*[22] (see the lower portion of Fig. 25). Instead of using the explanation of failure created during the blame assignment phase to post a set of learning goals that then direct the construction of a learning plan, the explanation can directly determine the choice of repair methods. System performance under both conditions can then be compared with Meta-AQUA under a no-learning situation.

Learning rates relative to a baseline no-learning condition are compared between the fully introspective and a semi-introspective version of Meta-AQUA. The independent variable that effects this change is the presence and influence of learning goals. The first experimental condition is referred to as the learning goal (LG) condition, and represents Meta-AQUA as described earlier in this article. Under the LG condition, the system builds a learning strategy. This construction is guided by the learning goals spawned by the Meta-XPs that explain the failure. Hence, this condition represents a loose coupling approach [15] between fault (failure cause) and repair (learning).

The second condition is called the random learning (RL) condition. Given an explanation by the blame assignment phase of the causes of failure, the system directly assigns calls to particular learning algorithms for each fault. The construction of the learning plan is then performed by a random ordering of these function calls, rather than by nonlinear planning to achieve the learning goals. The RL condition represents a tight coupling approach (i.e., the semi-introspective direct mapping from fault, or failure cause, to repair) that forgoes the use of learning goals and therefore skips the deciding what to learn stage.

The final condition is called the no learning (NL) condition in which Meta-AQUA performs story understanding, but if a failure exists, the system constructs no learning strategy. This condition represents the baseline performance from which both the LG and RL conditions can be compared. Holding all parameters constant except the independent

[22] It is semi-introspective because, although part of the introspective process has been removed, the introspective mechanics of blame assignment remain. Future research remains to test the performance with blame assignment removed and learning goals present.

variables, Meta-AQUA is given input from the Tale-Spin problem generator and the dependent variable is measured.

The dependent variable measures story understanding by rating the ability of the system to generate plausible explanations about key points in the story. The evaluation criterion in Meta-AQUA assigns credit as follows. For each anomalous or interesting input in a story, a point is given for posing a question, an additional point is given for providing any answer whatsoever, and a third point is assigned for answering what the researcher judges correct. The sum represents the dependent variable.

5.3. *The empirical data*

To serve as experimental trials and to minimize order effects, Tale-Spin generated six random sequences of Elvis-World stories. On each of these runs, Meta-AQUA processes a sequence three times, once for each experimental manipulation. The system begins all runs with the same initial conditions. For a given experimental condition, it processes all of the stories in the sequence while maintaining the learned knowledge between stories. At the end of the sequence, the system resets the BK. The input size for a run varies in length, but averages 27.67 stories per run.[23] The corpus for the six runs includes 166 stories, comprising a total of 4,884 sentences. The stories vary in size depending on the actions of the story and Tale-Spin's randomness parameters (e.g., the probability that the dog will accompany the officer), but average 29.42 sentences.

5.3.1. *Run number four*

Run number four is particularly interesting because the greatest number of learning interactions occurred in this set. The input to run four consisted of 24 stories as enumerated in Table 3. The stories contain a total of 715 sentences, and the average number of sentences per story is 29.8. Each numeric entry in Table 3 contains a triple of the form ⟨LG, RL, NL⟩. For example, the sixth column represents the number of learning episodes for each trial and for each condition. Note that the third element of each triple in this column is zero since learning is disabled in the NL condition. The fifth column (Question points) contains the values for the dependent variable. These values represent the sums of triples from the second, third and fourth columns (Posed questions, Answered questions and Correct answers, respectively). In this run, random drug busts occurred 11 times (5 with the canine squad and 6 with a lone police officer).

Examining the totals from Table 3, a number of trends can be discerned. The dependent variable (column 5, Question points) shows that Meta-AQUA's performance under the LG condition is significantly greater than the performance under the RL condition. In turn, Meta-AQUA performance in the RL condition far exceeded the performance under the NL condition. Alternatively, if only absolute performance (column 4, Correct answers) is considered, the differential is even greater. By this measure, the LG condition is more than

[23] The reason that each run varies in length is that, after generating around 600,000 gensyms, Meta-AQUA will use all available swap space on the Symbolics and thus inadvertently halt the underlying LISP. We then discard the story which is being processed at the time of the crash. The data from the remaining stories constitute the results of the run.

Table 3
Results from run number four

Story number (sentences)[a]	Questions posed (LG RL NL)	Answered questions (LG RL NL)	Correct answers (LG RL NL)	Question points (LG RL NL)	Learning episodes (LG RL NL)	Protagonist and problem[b]
01 (26)	1 1 1	0 0 0	0 0 0	1 1 1	1 1 0	Mom bored (balloon)
02 (19)	3 3 3	3 2 2	1 0 0	7 5 5	2 3 0	Mom bored (ball)
03 (38B)	1 1 1	0 0 0	0 0 0	1 1 1	1 1 0	Elvis jonesing
04 (51b)	1 1 1	1 0 0	1 0 0	3 1 1	0 1 0	Dad jonesing
05 (21)	1 1 1	1 0 0	1 0 0	3 1 1	0 1 0	Mom bored (ball)
06 (13)	1 1 1	1 0 0	1 0 0	3 1 1	0 1 0	Officer1 concerned
07 (13)	1 1 1	1 0 0	1 0 0	3 1 1	0 1 0	Dad bored (ball)
08 (21)	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	Dad thirsty
09 (44B)	2 2 2	2 1 1	1 0 0	5 3 3	1 2 0	Dad thirsty
10 (51B)	3 3 3	2 1 1	2 1 0	7 5 4	0 1 0	Dad bored (balloon)
11 (11)	2 2 1	1 1 1	1 0 0	4 3 2	1 2 0	Lynn bored (ball)
12 (3)	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	Officer1 concerned
13 (47b)	2 2 1	1 1 0	1 1 0	4 4 1	0 0 0	Mom thirsty
14 (15)	4 4 4	4 2 3	4 0 0	12 6 7	0 4 0	Mom bored (ball)
15 (28)	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	Lynn jonesing
16 (42B)	2 2 2	2 1 1	2 1 0	6 4 3	0 1 0	Dad jonesing
17 (45b)	2 2 1	1 1 0	1 1 0	4 4 1	0 0 0	Elvis jonesing
18 (21)	2 2 2	2 1 1	2 1 0	6 4 3	0 1 0	Officer1 concerned
19 (20)	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	Dad jonesing
20 (52b)	2 2 1	1 0 0	1 0 0	4 2 1	0 1 0	Dad bored (balloon)
21 (39b)	2 2 1	1 1 1	1 1 1	4 4 3	1 1 0	Lynn jonesing
22 (17)	2 2 2	2 1 1	2 0 0	6 3 3	0 2 0	Dad bored (ball)
23 (40B)	2 2 2	1 1 1	1 1 0	4 4 3	1 1 0	Elvis thirsty
24 (38b)	2 2 1	1 1 0	1 0 0	4 3 1	0 1 0	Mom bored (ball)
Total 715	38 38 32	28 15 13	25 7 1	91 60 46	8 26 0	

[a]The letter “B” means that the story contains an attempted drug bust by the police canine squad, whereas the letter “b” means that the officer entered the house alone to attempt a bust.
[b]Items in parentheses represent games played to dispel boredom.

three times the value of the RL condition, whereas, the performance of the NL condition is insignificant. By looking at column three, however, the numbers of questions answered in some way (right or wrong), are roughly equivalent in the RL and NL conditions, whereas the ratio of the LG condition to either of the other two is 2:1. Finally, the number of questions posed are virtually equal across all three conditions.

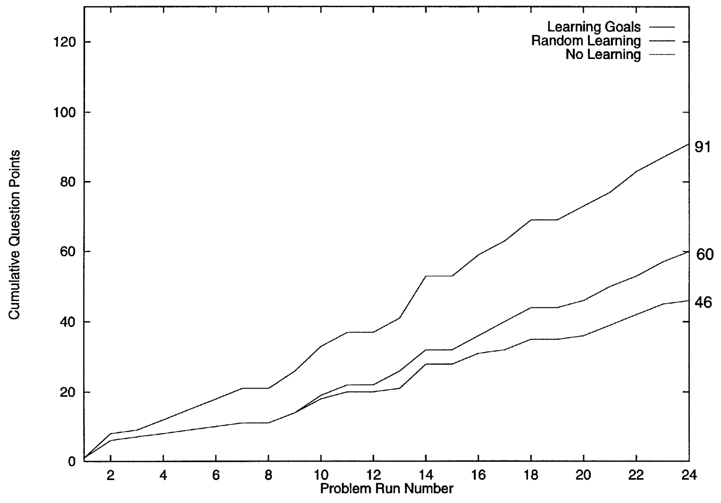

Fig. 26. Run 4, cumulative question points as a function of the number of problems.

In contrast to these differences, Meta-AQUA attempts to learn from failure more than three times as often under the RL condition as under the LG condition. That is, learning is more *effective* with learning goals than without. In the RL condition, learning does not increase performance as much as does the LG condition, while concurrently, it leads Meta-AQUA to expend more resources by increasing the amount of learning episodes. Thus, the system works harder and gains less under RL than under LG.

Fig. 26 graphs the accumulation of question points across trials (i.e., stories).[24] The behavior of the system as measured by the dependent variable is greatest under the LG condition, next best under RL, and worst under the NL condition. But, the trend does not hold for each trial. Fig. 27 shows raw scores indicating that the NL condition actually outperforms the RL condition on trial number 14. The reason for this effect is that under worse-case conditions, if the interactions present between learning methods are negative, the performance may actually degrade. As a result, randomly ordered learning may be worse than no learning at all.

The differences as a function of the independent variable are even more pronounced if only accuracy (the number of correct answers) is examined and partial credit ignored. Fig. 28 shows that under the RL condition, Meta-AQUA did not answer a question correctly until trial number 10, whereas under the NL condition, it did not perform correctly until trial 21. On the other hand, because under the LG condition the system learned a new explanation early in trial number 1, it was able to answer a question by trial number two.

5.3.2. Overall results

Table 4 summarizes the evaluation data from the six program runs. As is evident across all runs, the LG condition consistently outperforms the RL condition in the total cumulative question points. In turn, the RL condition outperforms the NL condition,

[24] The final extent of all three curves reach the value of the triple in the totals column for column five of Table 3.

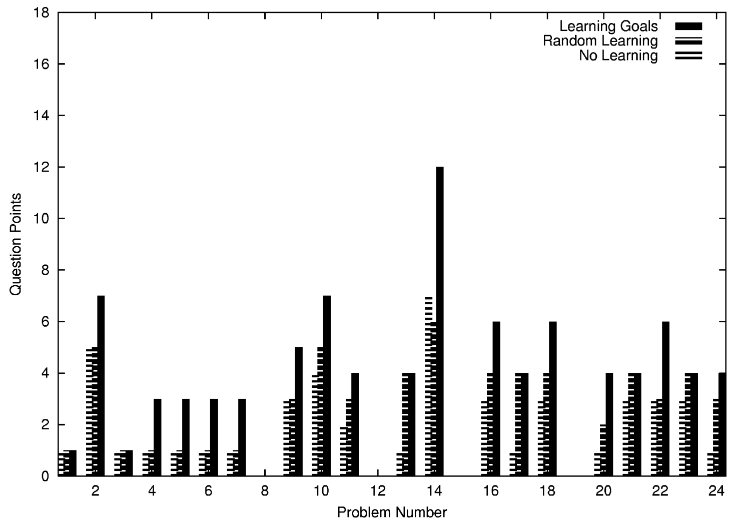

Fig. 27. Run 4, question points histogram.

despite the occasional poor performance due to negative interactions. As indicated by the standard deviations, the amount of differences between and within conditions exhibit high variability across runs.

Given these totals, the percent improvement for either learning condition over the NL base condition is simply the ratio of the difference in the base performance score and either score to the base score itself. Thus for run one, the ratio of the difference between the LG and NL conditions (35 points) to the NL condition (50 points) is 0.7, or 70%. Again, the improvement in performance for the LG condition is consistently higher than that of the RL condition. This difference is calculated in the final column. The differential is the percent improvement of the LG condition over the RL condition and is computed by the same measure as was the improvements in the individual learning conditions. That is, the differential is the ratio of the difference between the two improvements to the lower rate.[25] Thus, the differential between the LG rate of learning in run number one and that of the RL condition is the ratio of the difference (8 percentage points) to the RL percentage (62). Hence, the ratio is 0.129, or an improvement of nearly 13%.

Although the average differential between the two learning conditions (i.e., between fully-introspective and semi-introspective multistrategy learning) is more than 106% with a large standard deviation, this figure still overstates the difference. The expected gain in learning is more conservative. The differential between the average LG improvement (102.70) and the average RL improvement (65.67) is a 56.38% difference. In other words, across a number of input conditions, the use of learning goals to order and combine learning choices should show about 1.5 times the improvement in performance than will a straight mapping of faults to repairs when interactions are present.

[25] Note that this ratio can also be calculated as the difference between the performance scores of the learning conditions to the difference between the performance score of the RL and NL conditions. In other words, the ratio (LG – RL)/(RL – NL).

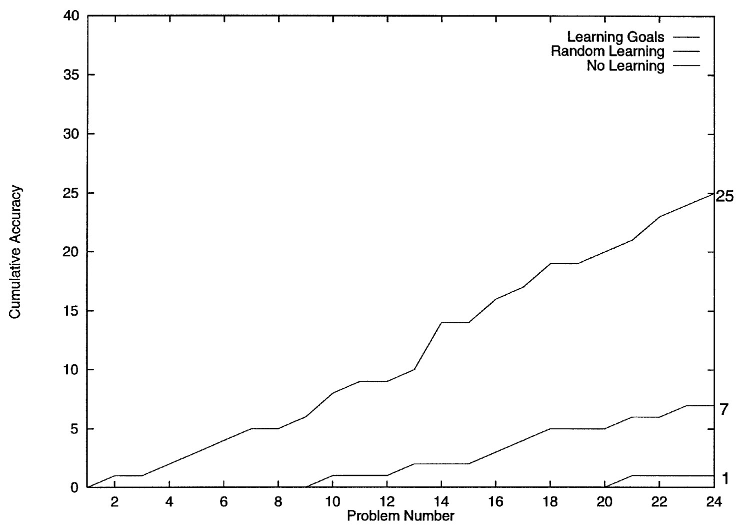

Fig. 28. Run 4, cumulative correct answers (accuracy) as a function of the number of problems.

Table 4
Summary of cumulative results

Run number[a]	Cumulative question points LG	RL	NL	% LG improved	% RL improved	Improvement differential
Run 1 (34)	85	81	50	70.00	62.00	12.90
Run 2 (30)	106	98	43	146.51	127.91	14.55
Run 3 (28)	120	102	60	100.00	70.00	42.86
Run 4 (24)	91	60	46	97.83	30.43	221.43
Run 5 (22)	57	49	27	111.11	81.48	36.36
Run 6 (28)	103	66	54	90.74	22.22	308.33
Averages	93.67	76.00	46.67	102.70	65.67	106.07
Std. Dev.	21.72	21.31	11.34	25.43	38.17	126.59

[a]Amounts in parentheses indicate total number of stories in each run.

6. Related research, contributions and discussion

This paper has presented a novel computational theory of learning. We have described learning as a deliberate planning process that actively seeks to improve knowledge in the context of particular performance tasks (for the examples presented here, in the context of story understanding). Learning is not incidental to the performance task, rather, it is a separate act of intelligence that requires knowledge, goals, and decisions. Given many choices as to the direction learning can assume, the strategic learner must construct a learning plan, given a reasoned analysis of its own knowledge and performance. To create such a plan, the learner must therefore be capable of some degree of introspection. Here we have outlined a knowledge representation, a set of algorithms, and a framework under which successful multistrategy learning can take place.

6.1. Related research

The work presented here builds on a very long research tradition in the case-based reasoning community on story understanding, explanation, memory, and inference. It has also been influenced by the extensive research efforts focused on meta-reasoning systems. This section examines a few of these sources to compare and contrast our work with the work that precedes us.

Our conceptualization of learning is consistent with both Michalski's [57] Inferential Learning Theory that decomposes a learning task into an input, the BK, and a learning goal and Carbonell [6] and Veloso's [97,98] emphasis on reasoning from a trace of the derivation of a solution rather than from solutions themselves. Unlike Michalski, we have stressed the role of the performance task in deciding what to learn and have used the metaphor of planning rather than inference when interpreting the learning task. Unlike Veloso and Carbonell who use a *derivational analogy trace* of past planning episodes to guide subsequent planning performance in the Prodigy/Analogy system, Meta-AQUA uses the information to guide subsequent learning. Although the effect on actual performance is therefore more indirect, the learning is more informed.

In general, our orientation is similar to many other approaches based on reasoning traces (e.g., [59,95]) or justification structures (e.g., [1,27,28]) to represent problem-solving performance and to other approaches that use characterizations of reasoning failures for blame assignment and multistrategy learning (e.g., [42,48,61,65,66,81,94]). Reasoning trace information has primarily been used for blame assignment during planning (e.g., [1,9]) and for speedup learning (e.g., [60]). A major difference between our approach and these approaches is our use of explicit representational structures (Introspective Meta-XPs) to represent classes of learning situations along with the types of learning needed in those situations.

Other types of knowledge may also be important in multistrategy or introspective learning systems. For example, Pazzani's [68] OCCAM system has generalized knowledge about physical causality that is used to guide multistrategy learning. In contrast, we propose specific knowledge about classes of learning situations that can be used to guide learning strategy selection and construction. The IULIAN system of Oehlmann, Edwards and Sleeman [64] maintains metacognitive knowledge in declarative introspection plans. The RFermi system [43] maintains goal and memory search information to represent knowledge about its memory performance. This information allows it to introspectively determine improvements in its search behavior. Freed's RAPTER system [16,32] uses three types of self-knowledge when learning. Records of variable bindings maintain an implicit trace of system performance, justification structures provide the knowledge of the kinds of cognitive states and events needed to explain the system's behavior, and transformation rules [8,34] describe how the mostly implementation-independent knowledge in justification structures corresponds to a particular agent's implementation. In the Meta-AQUA system, however, TMXPs maintain reasoning traces explicitly, and most implementation-dependent knowledge is avoided.

Our approach to using an analysis of reasoning failures to determine what needs to be learned is similar to Mooney and Ourston's [61] EITHER system, Park and Wilkins' [66] MINERVA program, the CASTLE system of Krulwich [9,47], Fox's [29,30] ROBBIE path planning system, and Stroulia's [93,94] Autognostic system, but with some important differences. We have focused on the use of meta-models for explicit representation of domain knowledge and of reasoning processes in an integrated, multistrategy reasoning and learning system. Unlike both Mooney and Ourston and Park and Wilkins, we have not assumed a single reasoning paradigm (logic-based deduction and rule-based expert systems, respectively) in terms of which failure situations and learning strategies are characterized. Rather, it is the architecture that provides a basis for a higher-level characterization, which in turn could be implemented in different ways depending on the reasoning paradigm. In fact, Meta-AQUA uses multiple reasoning methods, although the focus is on case-based reasoning.

Birnbaum and his colleagues [1] focuses on the process of blame assignment by backing up through justification structures, but do not emphasize the declarative representation of failure types. They explicitly model, however, the planner. They also explicitly model and reason about the intentions of a planner in order to find and repair the faults that underlie a planning failure (see [31]). Though much is shared between CASTLE and Meta-AQUA in terms of blame assignment (and to a great extent CASTLE is also concerned with deciding what to learn; see [46]), CASTLE does not use failure characterizations to formulate explicit learning goals nor does it construct a learning strategy in a deliberate manner within a multistrategy framework. The only other system to introspectively deliberate about the choice of a learning method is the ISM system [7]. ISM optimizes learning behavior dynamically and under reasoning failure or success, but the system chooses the best *single* learning algorithm, rather than composing a strategy from multiple algorithms. ISM does not therefore have to consider algorithm interactions.

Finally, our work focuses on reasoning failures, and not only on performance failures (in both the Collins et al. and Owens' cases, planning failures). Stroulia's approach focuses on a design stance characterization of the reasoner as a device, whereas our approach, as with the approach of Collins, Birnbaum, and their colleagues, takes a more intentional stance toward the reasoner. The analysis of [93], characterizing the ways in which such a device could fail, yields a taxonomy of failure types similar to ours. However, like the previous studies, she too does not use declarative characterizations of reasoning failures to formulate explicit learning goals. Furthermore, the CASTLE, Autognostic, ROBBIE, and RAPTER systems are all model-based in their introspective methods, whereas Meta-AQUA is XP-based (case-based). Finally, none of these systems create an explicit plan to learn in the space of changes to the BK. Despite these differences, we emphasize that the approaches enumerated in this section have much in common with the theory and implementation described here.

6.2. Contributions

This paper has presented a computational theory of deliberate learning that has explored the metaphor of nonlinear planning as a vehicle for constructing a learning strategy. The theory represents a general approach to knowledge-intensive learning, as supported by broad hand-coded evaluations [75], large-scale empirical experiments (reported here), and by previous psychological modelling studies [79].[26] This theory of multistrategy learning supports three major processing phases that represents an intelligent system's response to reasoning failure: blame assignment, deciding what to learn, and learning-strategy construction. Our contribution to blame assignment is that we have treated it as a goal-driven (deliberative) introspective process, rather than a data-driven knowledge-poor process (e.g., [3]). Just as case-based reasoners can represent actions in a story explicitly and then retrieve past cases with which to understand the story, a case-based introspector can represent the mental story understanding actions explicitly and then retrieve meta-explanations with which to understand its own comprehension failures. In support of this process, we have delineated the kinds of failures a reasoner can expect and the possible causes from which the learner can diagnose the problem. We also presented a knowledge structure to support the reasoning about these failure types. The Meta-XP representation encapsulates information about failure and the corresponding learning goals that if achieved can reduce the likelihood the failure will be repeated. In particular, we specified a novel representation of failures of omission such as forgetting. Because it represents the lack of a mental event, the solution had to be able to symbolize the truth value of a proposition in or out of the set of beliefs *with respect to* either the background knowledge or the system's foreground knowledge.

For the task of actually constructing a learning strategy, we have followed the metaphor of learning as planning. Our contribution is to show the surprising parallels that exist when one applies the metaphor strictly. We demonstrated that just as a nonlinear planner in the blocks world can devise a plan to achieve multiple goals of having blocks stacked on one another, the same planner can build a plan to achieve specific changes to its background knowledge. Instead of a plan consisting of physical movements of objects, a learning plan consists of sequenced calls to specific learning algorithms. To support these actions we presented a taxonomy of learning goals. Like the blocks world, the planner must be careful that actions performed in its plan do not interact negatively. The solution to these problems come from careful construction of learning operators that specify the preconditions and effects of the learning algorithms.

A central contribution of this paper, however, has been to define an intermediate stage of learning between the assignment of blame and the construction of a learning strategy. We postulated that for effective learning a system must deliberately focus on what needs to be learned. The experiments reported in this paper provide a number of results that support the hypothesis that this deciding what to learn stage of learning is necessary. Meta-AQUA expended more learning resources and induced less performance improvement without learning goals than it did under a condition that included them. Moreover, we have shown that because learning algorithms negatively interact, the arbitrary ordering of learning methods (i.e., as under the RL condition) can actually lead to worse system performance than no learning at all. Therefore, an explicit phase to decide exactly what to learn (i.e., to spawn learning goals or an equivalent mechanism) is necessary to avoid these interactions

[26] The multistrategy learning theory embodied in Meta-AQUA was the basis of a sister program called Meta-TS and has proven to be a sufficient model to cover actual human data in the domain of electronics troubleshooting behavior.

and to maintain effective learning in multistrategy environments. The paper also provided a novel quantitative measure with which to evaluate the comprehension process. As the dependent variable, this partial credit metric provides rewards for both posing questions autonomously and giving some type of answer, as well as for getting answers correct.

Because of the considerable computational overhead involved in maintaining a reasoning trace, performing blame assignment, spawning learning goals, and constructing a plan with which to pursue such goals, the benefits of using introspection must be substantial to justify the costs. Furthermore, under extremely complex situations or in informationally impoverished circumstances, deciding on an optimal learning goal is certainly intractable. In such situations, it may be more beneficial to proceed without further reasoning, rather than to attempt to understand the exact causes of the failure. Knowing when a learning task is worth pursuing is itself an important skill to master for an intelligent system. Identifying the most appropriate conditions for the use of an introspective approach is therefore a desirable research goal. To establish only that introspection facilitates learning and that the model of introspection has some quality of reasonableness is not fully satisfactory. Further inquiry into these conditions is left for future research.

In the interim, a potential heuristic for deciding when to use an introspective approach is to qualitatively ascertain whether or not interactions between learning mechanisms available to the learner exist. If they exist, then the approach should be applied, otherwise a more reflexive approach is licensed. In speculation, another potential heuristic for determining that introspection is a win is to use a threshold for the number of failure symptoms above which introspection will not be attempted. Through experimentation, this threshold number should be obtained empirically given a distribution of known problem types and a random selection of problems from the distribution. The identification of such heuristics will enable the practical use of introspective methods in systems that cannot afford to squander precious resources with intractable computation.

Acknowledgement

This research was supported by NSF under contract IRI-9009710, by AFOSR under contract #F49620-94-1-0092, and by the Georgia Institute of Technology. We thank Tucker Balch, Kurt Eiselt, Karen Haigh, Janet Kolodner, Kenny Moorman, Mimi Recker, Juan Carlos Santamaria, Tony Simon, Austin Tate, Manuela Veloso and the anonymous reviewers for comments on various drafts of this paper. We also wish to gratefully acknowledge Mark Devaney's assistance in reprogramming Tale-Spin.

References

[1] L. Birnbaum, G. Collins, M. Freed, B. Krulwich, Model-based diagnosis of planning failures, in: Proc. AAAI-90, Boston, MA, AAAI Press, Menlo Park, CA, 1990, pp. 318–323.

[2] D.G. Bobrow, A. Collins (Eds.), Representation and Understanding: Studies in Cognitive Science, Academic Press, New York, 1975.

[3] L.B. Booker, D.E. Goldberg, J.H. Holland, Classifier systems and genetic algorithms, Artificial Intelligence 40 (1989) 235–282.

[4] J.G. Carbonell, Subjective Understanding: Computer Models of Belief Systems, UMI Research Press, Ann Arbor, MI, 1981.

[5] J.G. Carbonell, Learning by analogy: Formulating and generalizing plans from past experience, in: R.S. Michalski, J.G. Carbonell, T.M. Mitchell (Eds.), Machine Learning I: An Artificial Intelligence Approach, Morgan Kaufmann, Los Altos, CA, 1983, pp. 137–161.

[6] J.G. Carbonell, Derivational analogy: A theory of reconstructive problem solving and expertise acquisition, in: R. Michalski, J. Carbonell, T. Mitchell (Eds.), Machine Learning II: An Artificial Intelligence Approach, Morgan Kaufmann, San Mateo, CA, 1986, pp. 371–392.

[7] J. Cheng, Management of speedup mechanisms in learning architectures, Ph.D. Thesis, Technical Report No. CMU-CS-95–112, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA, 1995.

[8] G. Collins, Plan creation: Using strategies as blueprints, Ph.D. Thesis, Technical Report No. 599, Yale University, Department of Computer Science, New Haven, CT, 1987.

[9] G. Collins, L. Birnbaum, B. Krulwich, M. Freed, The role of self-models in learning to plan, in: A. Meyrowitz (Ed.), Machine Learning: Induction, Analogy and Discovery, Kluwer Academic, Boston, MA, 1993.

[10] M.T. Cox, Machines that forget: Learning from retrieval failure of mis-indexed explanations, in: A. Ram and K. Eiselt (Eds.), Proc. 16th Annual Conference of the Cognitive Science Society, Lawrence Erlbaum Associates, Hillsdale, NJ, 1994, pp. 225–230.

[11] M.T. Cox, Representing mental events (or the lack thereof), in: M.T. Cox, M. Freed (Eds.), Proc. 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms, Technical Report No. SS-95-05, AAAI Press, Menlo Park, CA, 1995, pp. 22–30.

[12] M.T. Cox, An empirical study of computational introspection: Evaluating introspective multistrategy learning in the Meta-AQUA system, in: R.S. Michalski, J. Wnek (Eds.), Proc. 3rd International Workshop on Multistrategy Learning, AAAI Press/MIT Press, Menlo Park, CA, 1996, pp. 135–146.

[13] M.T. Cox, Introspective multistrategy learning: Constructing a learning strategy under reasoning failure, Ph.D. Thesis, Technical Report No. GIT-CC-96-06, Georgia Institute of Technology, College of Computing, Atlanta, GA, 1996; ftp://ftp.cc.gatech.edu/pub/ai/ram/git-cc-96-06.html.

[14] M.T. Cox, An explicit representation of reasoning failures, in: D.B. Leake, E. Plaza (Eds.), Case-Based Reasoning Research and Development: Second International Conference on Case-Based Reasoning, Springer, Berlin, 1997, pp. 211–222.

[15] M.T. Cox, 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, Springer, Berlin, 1997, pp. 425–434.

[16] M.T. Cox, M. Freed, Using knowledge from cognitive behavior to learn from failure, in: J.W. Brahan, G.E. Lasker (Eds.), Proc. 7th International Conference on Systems Research, Informatics and Cybernetics, Vol. 2, Advances in Artificial Intelligence—Theory and Application II, The International Institute for Advanced Studies in Systems Research and Cybernetics, Windsor, Ontario, Canada, 1994, pp. 142–147.

[17] M.T. Cox, A. Ram, Using introspective reasoning to select learning strategies, in: R.S. Michalski, G. Tecuci (Eds.), Proc. First International Workshop on Multistrategy Learning, George Mason University, Artificial Intelligence Center, Washington, DC, 1991, pp. 217–230.

[18] M.T. Cox, A. Ram, An explicit representation of forgetting, in: J.W. Brahan, G.E. Lasker (Eds.), Proc. 6th International Conference on Systems Research, Informatics and Cybernetics, Vol. 2, Advances in Artificial Intelligence—Theory and Application, The International Institute for Advanced Studies in Systems Research and Cybernetics, Windsor, Ontario, Canada, 1992, pp. 115–120.

[19] M.T. Cox, A. Ram, Multistrategy learning with introspective meta-explanations, in: D. Sleeman, P. Edwards (Eds.), Machine Learning: Proc. 9th International Conference, Morgan Kaufmann, San Mateo, CA, 1992, pp. 123–128.

[20] M.T. Cox, A. Ram, Failure-driven learning as input bias, in: A. Ram, K. Eiselt (Eds.), Proc. 16th Annual Conference of the Cognitive Science Society, Lawrence Erlbaum Associates, Hillsdale, NJ, 1994, pp. 231–236.

[21] M.T. Cox, A. Ram, Interacting learning-goals: Treating learning as a planning task, in: J.-P. Haton, M. Keane, M. Manago (Eds.), Advances in Case-Based Reasoning: Second European Workshop, EWCBR-94, Springer, Berlin, 1995, pp. 60–74.

[22] M.T. Cox, A. Ram, On the intersection of story understanding and learning, in: A. Ram, K. Moorman (Eds.), Computational Models of Reading and Understanding, MIT Press/Bradford Books, Cambridge, MA, 1999, pp. 397–433.
[23] M.T. Cox, M.M. Veloso, Goal transformations in continuous planning, in: M. desJardins (Ed.), Proc. 1998 AAAI Fall Symposium on Distributed Continual Planning, AAAI Press/MIT Press, Menlo Park, CA, 1998, pp. 23–30.
[24] R. Cullingford, Script application: Computer understanding of newspaper stories, Ph.D. Thesis, Technical Report No. 116, Yale University, Department of Computer Science, New Haven, CT, 1978.
[25] R. Cullingford, Micro SAM, in: R.C. Schank, C. Riesbeck (Eds.), Inside Computer Understanding: Five Programs Plus Miniatures, Lawrence Erlbaum Associates, Hillsdale, NJ, 1981, pp. 120–135.
[26] G. DeJong, R. Mooney, Explanation-based learning: An alternative view, Machine Learning 1 (2) (1986) 145–176.
[27] J. de Kleer, J. Doyle, G.L. Steele, G.L. Sussman, Explicit control of reasoning, SIGPLAN Notices 12 (1977).
[28] J. Doyle, A truth maintenance system, Artificial Intelligence 12 (1979) 231–272.
[29] S. Fox, Introspective learning for case-based planning, Ph.D. Thesis, Technical Report No. TR462, Indiana University, Department of Computer Science, Bloomington, IN, 1995.
[30] S. Fox, D. Leake, Modeling case-based planning for repairing reasoning failures, in: M.T. Cox, M. Freed (Eds.), Proc. 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms, Technical Report No. SS-95-05, AAAI Press, Menlo Park, CA, 1995, pp. 31–38.
[31] M. Freed, B. Krulwich, L. Birnbaum, G. Collins, Reasoning about performance intentions, in: Proc. 14th Annual Conference of the Cognitive Science Society, Lawrence Erlbaum Associates, Hillsdale, NJ, 1992, pp. 7–12.
[32] M. Freed, G. Collins, Learning to cope with task interactions, in: A. Ram, M. desJardins (Eds.), Proc. 1994 AAAI Spring Symposium on Goal-Driven Learning, Technical Report No. SS-94-02, AAAI Press, Menlo Park, CA, 1994, pp. 28–35.
[33] S. Ghosh, J. Hendler, S. Kambhampati, B. Kettler, UM Nonlin [a Common Lisp implementation of A. Tate's Nonlin planner], 1992; FTP: Hostname: cs.umd.edu Directory: /pub/nonlin Files: nonlin-files.tar.Z.
[34] K.J. Hammond, Case-Based Planning: Viewing Planning as a Memory Task, Academic Press, San Diego, CA, 1989.
[35] K.J. Hammond, Explaining and repairing plans that fail, Artificial Intelligence 45 (1990) 173–228.
[36] K.J. Hammond, T. Converse, M. Marks, C. Seifert, Opportunism and learning, in: J.L. Kolodner (Ed.), Case-Based Learning, Kluwer Academic, Boston, MA, 1993, pp. 85–115.
[37] F. Hayes-Roth, Using proofs and refutations to learn from experience, in: R.S. Michalski, J.G. Carbonell, T.M. Mitchell (Eds.), Machine Learning I: An Artificial Intelligence Approach, Morgan Kaufmann, Los Altos, CA, 1983, pp. 221–240.
[38] J.R. Hobbs, Coherence and co-reference, Cognitive Sci. 3 (1) (1979) 67–82.
[39] J.R. Hobbs, M. Stickel, D. Appelt, P. Martin, Interpretation as abduction, Artificial Intelligence 63 (1–2) (1993) 69–142.
[40] L.E. Hunter, Planning to learn, in: Proc. 12th Annual Conference of the Cognitive Science Society, Lawrence Erlbaum Associates, Hillsdale, NJ, 1990, pp. 261–276.
[41] L. Ihrig, S. Kambhampati, An explanation-based approach to improve retrieval in case-based planning, in: M. Ghallab, A. Milani (Eds.), New Directions in Planning, IOS Press, Amsterdam, 1996, pp. 395–406.
[42] A. Kass, Developing creative hypotheses by adapting explanations, Ph.D. Thesis, Northwestern University, The Institute for the Learning Sciences, Evanston, IL, 1990.
[43] A.C. Kennedy, Using a domain-independent introspection mechanism to improve memory search, in: M.T. Cox, M. Freed (Eds.), Proc. 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms, Technical Report No. SS-95-05, AAAI Press, Menlo Park, CA, 1995, pp. 72–78.
[44] J.L. Kolodner, Capitalizing on failure through case-based inference, in: Proc. 9th Annual Conference of the Cognitive Science Society, Lawrence Erlbaum Associates, Hillsdale, NJ, 1987, pp. 715–726.
[45] J.L. Kolodner, Case-Based Reasoning, Morgan Kaufmann, San Mateo, CA, 1993.
[46] B. Krulwich, Determining what to learn in a multi-component planning system, in: Proc. 13th Annual Conference of the Cognitive Science Society, Lawrence Erlbaum Associates, Hillsdale, NJ, 1991, pp. 102–107.

[47] B. Krulwich, Flexible learning in a multicomponent planning system, Ph.D. Thesis, Technical Report No. 46, Northwestern University, The Institute for the Learning Sciences, Evanston, IL, 1993.

[48] D. Leake, Evaluating Explanations: A Content Theory, Lawrence Erlbaum Associates, Hillsdale, NJ, 1992.

[49] M. Lebowitz, Experiments with incremental concept formation: UMIMEM, Machine Learning 2 (1987) 103–138.

[50] W. Lehnert, M.G. Dyer, P. Johnson, C. Yang, S. Harley, BORIS—An experiment in in-depth understanding of narratives, Artificial Intelligence 20 (1) (1983) 15–62.

[51] C.E. Martin, Micro DMAP, in: C.K. Riesbeck, R.C. Schank (Eds.), Inside Case-Based Reasoning, Lawrence Erlbaum Associates, Hillsdale, NJ, 1989, pp. 353–372.

[52] C.E. Martin, Direct memory access parsing, Ph.D. Thesis, Technical Report No. 93-07, University of Chicago, Department of Computer Science, Chicago, IL, 1990.

[53] J. McCarthy, Programs with common sense, in: Symposium Proceedings on Mechanisation of Thought Processes, Vol. 1, Her Majesty's Stationary Office, London, 1959, pp. 77–84.

[54] J. McCarthy, Making robots conscious of their mental states, in: M.T. Cox, M. Freed (Eds.), Proc. 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms, Technical Report No. SS-95-05, AAAI Press, Menlo Park, CA, 1995, pp. 89–96.

[55] D. McDermott, A general framework for reason maintenance, Technical Report No. 691, Yale University, Department of Computer Science, New Haven, CT, 1989; also: Artificial Intelligence 50 (1991) 289–329.

[56] J. Meehan, Talespin, in: R.C. Schank, C. Riesbeck (Eds.), Inside Computer Understanding: Five Programs Plus Miniatures, Lawrence Erlbaum Associates, Hillsdale, NJ, 1981, pp. 197–258.

[57] R.S. Michalski, Inferential theory of learning: Developing foundations for multistrategy learning, in: R.S. Michalski, G. Tecuci (Eds.), Machine Learning IV: A Multistrategy Approach, Morgan Kaufmann, San Francisco, CA, 1994, pp. 3–61.

[58] M.L. Minsky, Matter, mind, and models, in: Proc. International Federation of Information Processing Congress, Vol. 1, 1965, pp. 45–49 (Original work from 1954).

[59] S. Minton, Learning Search Control Knowledge: A Explanation-Based Approach, Kluwer Academic, Boston, MA, 1988.

[60] T. M. Mitchell, R. Keller, S. Kedar-Cabelli, Explanation-based generalization: A unifying view, Machine Learning 1 (1) (1986) 47–80.

[61] R. Mooney, D. Ourston, A multistrategy approach to theory refinement, in: R.S. Michalski, G. Tecuci (Eds.), Machine Learning IV: A Multistrategy Approach, Morgan Kaufmann, San Francisco, CA, 1994, pp. 141–164.

[62] K. Moorman, A. Ram, Creativity in reading: Understanding novel concepts, in: A. Ram, K. Moorman (Eds.), Computational Models of Reading and Understanding, MIT Press/Bradford Books, Cambridge, MA, 1999, pp. 359–395.

[63] A. Newell, H.A. Simon, GPS: A program that simulates human thought, in: E.A. Feigenbaum, J. Feldman (Eds.), Computers and Thought, McGraw Hill, New York, 1963, pp. 279–293 (Original work published 1961).

[64] R. Oehlmann, P. Edwards, D. Sleeman, Introspection planning: Representing metacognitive experience, in: M.T. Cox, M. Freed (Eds.), Proc. 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms, Technical Report No. SS-95-05, AAAI Press, Menlo Park, CA, 1995, pp. 102–110.

[65] C. Owens, Indexing and retrieving abstract planning knowledge, Ph.D. Thesis, Yale University, Department of Computer Science, New Haven, CT, 1990.

[66] Y.T. Park, D.C. Wilkins, Establishing the coherence of an explanation to improve refinement of an incomplete knowledge base, in: Proc. AAAI-90, Boston, MA, AAAI Press, Menlo Park, CA, 1990, pp. 318–323.

[67] M. Pazzani, Learning fault diagnosis heuristics from device descriptions, in: Y. Kodratoff, R.S. Michalski (Eds.), Machine Learning III: An Artificial Intelligence Approach, Morgan Kaufmann, San Mateo, CA, 1990, pp. 214–234.

[68] M. Pazzani, Learning causal patterns: Making a transition from data-driven to theory-driven learning, in: R. Michalski, G. Tecuci (Eds.), Machine Learning IV: A Multistrategy Approach, Morgan Kaufmann, San Francisco, CA, 1994, pp. 267–293.

[69] M. Pollack, A model of plan inference that distinguishes between the beliefs of actors and observers, in: Proc. 24th Annual Meeting of the Association for Computational Linguistics, 1986, pp. 207–214.

[70] M.R. Quillian, Semantic memory, in: M.L. Minsky (Ed.), Semantic Information Processing, MIT Press, Cambridge, MA, 1968, pp. 227–270.
[71] A. Ram, Knowledge goals: A theory of interestingness, in: Proc. 12th Annual Conference of the Cognitive Science Society, Lawrence Erlbaum Associates, Hillsdale, NJ, 1990, pp. 206–214.
[72] A. Ram, A theory of questions and question asking, J. Learning Sciences 1 (3–4) (1991) 273–318.
[73] A. Ram, Indexing, elaboration and refinement: Incremental learning of explanatory cases, Machine Learning 10 (1993) 201–248.
[74] A. Ram, AQUA: Questions that drive the understanding process, in: R.C. Schank, A. Kass, C.K. Riesbeck (Eds.), Inside Case-Based Explanation, Lawrence Erlbaum Associates, Hillsdale, NJ, 1994, pp. 207–261.
[75] A. Ram, M.T. Cox, Introspective reasoning using meta-explanations for multistrategy learning, in: R.S. Michalski, G. Tecuci (Eds.), Machine Learning IV: A Multistrategy Approach, Morgan Kaufmann, San Francisco, CA, 1994, pp. 349–377.
[76] A. Ram, L. Hunter, The use of explicit goals for knowledge to guide inference and learning, Applied Intelligence 2 (1) (1992) 47–73.
[77] A. Ram, D. Leake, Evaluation of explanatory hypotheses, in: Proc. 13th Annual Conference of the Cognitive Science Society, Lawrence Erlbaum Associates, Hillsdale, NJ, 1991, pp. 867–871.
[78] A. Ram, D. Leake, Learning, goals, and learning goals, in: A. Ram, D. Leake (Eds.), Goal-Driven Learning, MIT Press/Bradford Books, Cambridge, MA, 1995, pp. 1–37.
[79] A. Ram, S. Narayanan, M.T. Cox, Learning to trouble-shoot: Multistrategy learning of diagnostic knowledge for a real-world problem solving task, Cognitive Sci. 19 (1995) 289–340.
[80] M. Ranney, P. Thagard, Explanatory coherence and belief revision in naive physics, in: Proc. 10th Annual Conference of the Cognitive Science Society, Lawrence Erlbaum Associates, Hillsdale, NJ, 1988, pp. 426–432.
[81] M.A. Redmond, Learning by observing and understanding expert problem solving, Ph.D. Thesis, Technical Report No. GIT-CC-92/43, Georgia Institute of Technology, College of Computing, Atlanta, GA, 1992.
[82] W.S. Reilly, J. Bates, Natural negotiation for believable agents, Technical Report No. CMU-CS-95-164, Carnegie Mellon University, Computer Science Department, Pittsburgh, PA, 1995.
[83] C.K. Riesbeck, R.C. Schank (Eds.), Inside Case-Based Reasoning, Lawrence Erlbaum Associates, Hillsdale, NJ, 1989.
[84] E.D. Sacerdoti, A Structure for Plans and Behavior, Elsevier, New York, 1977.
[85] R.C. Schank, Conceptual Information Processing, North-Holland, Amsterdam, 1975.
[86] R.C. Schank, Interestingness: Controlling inferences, Artificial Intelligence 12 (1979) 273–297.
[87] R.C. Schank, Dynamic Memory: A Theory of Reminding and Learning in Computers and People, Cambridge University Press, Cambridge, MA, 1982.
[88] R.C. Schank, Explanation Patterns: Understanding Mechanically and Creatively, Lawrence Erlbaum Associates, Hillsdale, NJ, 1986.
[89] R.C. Schank, A. Kass, C.K. Riesbeck, Inside Case-Based Explanation, Lawrence Erlbaum Associates, Hillsdale, NJ, 1994.
[90] R.C. Schank, C.C. Owens, Understanding by explaining expectation failures, in: R.G. Reilly (Ed.), Communication Failure in Dialogue and Discourse, Elsevier, New York, 1987.
[91] R.C. Schank, D. Leake, Creativity and learning in a case-based explainer, Artificial Intelligence 40 (1–3) (1989) 353–385.
[92] P.J. Schwanenflugel, W.V. Fabricius, C.R. Noyes, K.D. Bigler, J.M. Alexander, The organization of mental verbs and folk theories of knowing, J. Memory and Language 33 (1994) 376–395.
[93] E. Stroulia, Failure-driven learning as model-based self-redesign, Ph.D. Thesis, Technical Report No. GIT-CC-95-38, Georgia Institute of Technology, College of Computing, Atlanta, GA, 1994.
[94] E. Stroulia, A. Goel, Functional representation and reasoning for reflective systems, J. Appl. Intelligence 9 (1) (1995) 101–124.
[95] G.J. Sussman, A Computer Model of Skill Acquisition, American Elsevier, New York, 1975.
[96] A. Tate, Project planning using a hierarchic nonlinear planner, Technical Report No. 25, University of Edinburgh, Department of Artificial Intelligence, Edinburgh, UK, 1976.
[97] M.M. Veloso, Planning and Learning by Analogical Reasoning, Springer, Berlin, 1994.
[98] M.M. Veloso, J.G. Carbonell, Case-based reasoning in PRODIGY, in: R.S. Michalski, G. Tecuci (Eds.), Machine Learning IV: A Multistrategy Approach, Morgan Kaufmann, San Francisco, CA, 1994, pp. 523–548.

[99] R. Wilensky, Planning and Understanding: A Computational Approach to Human Reasoning, Addison-Wesley, Reading, MA, 1983.

[100] D.E. Wilkins, Domain independent planning: Representation and plan generation, Artificial Intelligence 22 (1984) 269–301.

[101] D. Zeng, K. Sycara, Benefits of learning in negotiation, in: Proc. 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference, Morgan Kaufmann, San Francisco, CA, 1997, pp. 36–41.