

HILBERT & PATRIC: Hybrid Intelligent Agent Technology for Teaching Context-Independent Reasoning

Paul Bello

Rensselaer AI and Reasoning (RAIR) Laboratory
Department of Computer Science
Department of Cognitive Science
Rensselaer Polytechnic Institute (RPI)
Troy NY 12180 USA
bellop@rpi.edu

Selmer Bringsjord

Rensselaer AI and Reasoning (RAIR) Laboratory (Director)
Professor of Logic, Cognitive Science, Computer Science
Department of Cognitive Science (Chair)
Department of Computer Science
Rensselaer Polytechnic Institute (RPI)
Troy NY 12180 USA
selmer@rpi.edu

Abstract

There is a disturbing paradox at the heart of contemporary American education: As this education turns more 'electronic,' we are moving away from the one kind of learning that we know to be most effective, namely, one-on-one instruction. As the need for good teachers at the university level continues to grow, we see this paradox intensifying. A specific exacerbating force is that, as the data tells us, as educators, we are not producing students able to successfully employ context-independent reasoning (CIDR) in technical domains. Our future scientists and engineers have been shown to lack this fundamental tool of their trade. The fact is, teaching students to be good CIDR reasoners requires the professor to develop a one-on-one relationship with each student, with a keen eye on how each applies their own strategies for proof construction. With this in mind, we envision the automated logic instructor as adaptable, and fully available to each student, at all times. This is obviously not possible with human instruction, but our preliminary work suggests that our vision is capable of being realized in the digital domain: We are developing a suite of intelligent agents, including PATRIC and HILBERT that marries the cutting edge in AI-driven tutoring with the state-of-the-art in proof construction courseware.

Keywords

Intelligent tutoring systems, Context-independent reasoning, Hybrid intelligent systems, Neural networks, Agent-driven instruction

Introduction

The advent of the portable digital computer and the Web has revolutionized the way we work and play, and the way we learn and teach. Today's grade-sheets are being kept online, and homework assignments pile up in the inbox, not on the desk. To further complicate the issue, a growing college-age population (at least in the US) is causing that pile to become larger every year. Like many other instructors in computer science, logic, and related areas, we have started to feel the pressure. We thus seek technology that frees up our time as teachers so that we are able to discuss deep issues with students on an individual basis.

Context-independent reasoning is a vital skill for success in today's high-tech society. Despite the fact that this skill is supposedly taught from high school through college, there is a remarkable dearth of good context-independent reasoners in our college classrooms. The natural inference is that we're just not doing a good enough job teaching our students how to reason in an abstract, formal manner.

It has been repeatedly shown that one-on-one tutoring is much more pedagogically effective than traditional lecture-based instruction (Anderson et al., 1995; Cohen et al., 1982). Many researchers have seen this as an invitation to create Intelligent Tutoring Systems (ITS): machines able to teach in one-on-one fashion. However, a

tutoring system specifically able to teach context-independent reasoning has yet to be developed and fielded in the classroom.

This paper seeks to define our objectives in broad strokes, while presenting results that have already been obtained in classroom studies. We propose a two-pronged attack on this problem: instantiated intelligent agent technologies called PATRIC & HILBERT. These two agents lie at the very heart of our intelligent tutoring initiative. PATRIC is an agent capable of real-time interaction with student users. Individualized interaction is accomplished through a set of data generated via HILBERT's analysis of students' homework assignments. Our preliminary work has shown that an agent-driven presentation of material achieved better results on a post-test among an experimental group of subjects than those who learned the material in standard lecture format. Our motivation is to deliver this brand of agent-driven instruction to the masses without the dependence on a human instructor to facilitate learning.

The Dilemma

The initiative at Carnegie Mellon University to develop an automatic tutoring system was the brainchild of Scheines and Sieg (Scheines and Sieg, 1990). The system is built on top of the Valid theorem proving utility developed by Patrick Suppes at Stanford University (Suppes 1981). CMPT (Carnegie Mellon Proof Tutor) was intended to replace a traditional first course in logic, and met with a surprising amount of success. These preliminary successes, as important as they were, don't add up to an immersive tutoring experience. CMPT can only be used with the propositional calculus (a subset of first order logic), and to date hasn't been upgraded to handle FOL. The proof is made much more difficult to read using the proof tree representation. There is a considerable lack of seamless interactivity.

While the goal remains to teach a broad student base how to assemble proofs in first order logic, most of the brainpower of these programs is in the automatic theorem prover that works behind the scenes. This does little to make the software easy to use, or the visual formalisms that it uses more palatable to the eye. In the case of CMPT, the preferred representation is a Fitch-style natural deduction proof (which is what we'd like to use as well), using windowing to encapsulate assumptions and their corresponding results in the proof. Unfortunately, even though CMPT can complete a proof from any given step in the reasoning process, this is hardly ground to deem it a tutor, but certainly makes it an interesting candidate from the standpoint of automated theorem proving.

The Solution: Agent Driven Logic Instruction

Intelligent agent technology may well offer the tools and techniques needed to unravel the mystery of how to provide users with a tutoring experience that is truly 'immersive'. Our agents (HILBERT and PATRIC) provide our tutoring architecture with a wealth of information that has been previously unavailable from logic tutors, including a rich set of beliefs, a multi-modal set of communication possibilities (both textual and voice-interactive), and an intuitive interface for constructing and evaluating proofs.

Our first efforts have been on the construction of an intelligent agent capable of offering limited real-time advice on a one-on-one basis. As such, an agent of this type is responsible for monitoring the progress of the user, and for providing advice and hints when requested in an interactive session. Since all of our work is based on the intelligent agent paradigm, and since, in particular, intelligent agents come in many varieties, a brief review of the paradigm is necessary. We begin with the textbook definition, given in *Artificial Intelligence: A Modern Approach (AIMA)* (Russell, Norvig 1994 p. 31):

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors. A human agent has eyes, ears, and other organs for sensors, and hands, legs, mouth, and other body parts for effectors. A robotic agent substitutes cameras and infrared range finders for sensors and various motors for effectors. A software agent has encoded bit strings as its percepts and actions. Let's have a look at two different types of agents and what they can do for us.

Reflex Agents

In the preliminary stages of this project, our agent manifested itself as a simple reflex agent with some enhancements that we will proceed to describe in detail. A reflex agent (Figure 1) is simply a collection of *condition-action* rules that have the form: if *condition* then *action*. These rules govern the behavior of the system with respect to its percepts and constitute the agent's abstract view of the world. For our purposes, we need an agent that is able to make judgments and give advice based on the current information it has gathered thus far about the student via examination of his or her submitted work. These 'beliefs' are useful in tailoring advice towards the style of learning that correlates with high performance on the part of the user. Unfortunately, reflex agents aren't very useful in terms of making judgments based on temporal factors. Luckily enough for us, our augmented agent relies on beliefs that are built up over time to construct its knowledge-base. A simple algorithmic sketch of such an agent is shown in Figure 2. This basic procedure was implemented successfully in our first experiments.

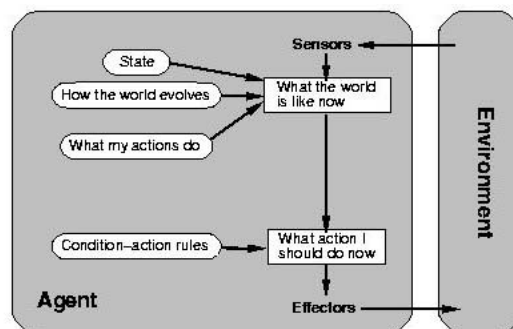


Figure 1. Reflex Agent with State (from AIMA)

```
function REFLEX-AGENT-WITH-STATE(percept) returns action
  state ← state, a description of the current world state
  rules ← rules, a set of condition-action rules

  state ← UPDATE-STATE(state, percept)
  rule ← RULE-MATCH(state, rules)
  action ← RULE-ACTION[rule]
  state ← UPDATE-STATE(state, action)
  return action
```

Figure 2. Algorithm for Reflex Agent with State (from AIMA)

Goal-Based Agents

At the heart of all tutoring systems lies the *a priori* knowledge of expert(s) in the chosen domain. A tutoring system for logic will often have explicitly coded solutions for each exercise that a student is presented with. We believe that an agent should be able to take the current state of a partially completed proof, and generate a maximally informative sequence of steps (in accordance with knowledge about the user) to eventually satisfy all of the goals. The goal-driven functionality embedded in our agent subsumes both traditional planning agents and utility-based agents. Our PATRIC (Pedagogical Agent for Teaching Reasoning Independent of Context), an augmented reflex-agent architecture, is shown below in Figure 3.

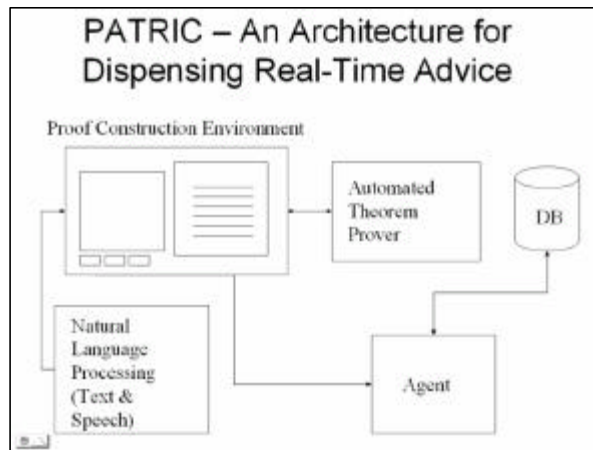


Figure 3. Augmented Reflex Agent for Real-Time Advice

The Microsoft Agent

Given that our architecture has all of these capabilities; we needed to find a robust embodiment to act as our vehicle for disseminating information. The vehicle of choice in this case turned out to be a Microsoft Agent. A Microsoft Agent is a programmable desktop caricature that is capable of speaking, moving, gesturing, receiving spoken language, and responding. This relatively rich set of behaviors is a more than suitable way to embody the agent architecture we've mentioned previously. Our agent of choice is Robby the Robot. Robby is shown in Figure 4. The agent is run by Visual Basic code embedded in the web pages that our tutoring software displays in its browser window.



Figure 4. Robby, our Hardworking Assistant

We are using the full suite of Microsoft Agent functionality in the algorithm design. Robby will respond both to voice commands and to a HELP button on the application. The infrastructure of the agent is based on the algorithm shown in figure 2, and provides the user with one or more steps in the progression of the proof, depending on how much help is requested.

Our First Experiment

To judge the efficacy of our agent, we carried out an experiment comparing students who did not have a physically present instructor with our artificial agent. The agent was used to teach a specific topic, proof by contradiction (or *reductio ad absurdum*), to students from the RPI *Introduction to Logic* class. After dividing volunteers into two groups matched for ability in logic by a pre-test (the 1998 version, differing only in date, is available at <http://www.rpi.edu/~faheyj2/SB/INTLOG/pre-test.f98.pdf>), we offered the students in the experimental group an hour-long course in the use of the software, without presenting to them any content. We then gave each group simultaneous instruction: the experimental group viewed the interactive software, the control received instruction as normal from their professor.

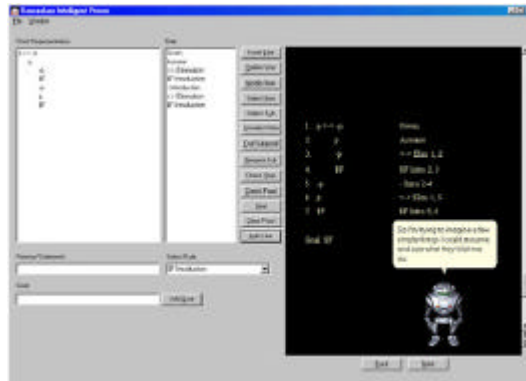


Figure 5. The Rensselaer Intelligent Prover in Action

Following the instruction, we gave each participant a post-test consisting of a logic problem that required understanding and applying the concept discussed in the lessons (again, proof by contradiction). Three of six students in the experimental group received full credit; only one of seven did so from the control, with another earning partial credit. Relatively low attendance rates complicated the statistical interpretation somewhat, but the difference was reported at a significance of .092, indicating that it was quite unlikely that in-person instruction was better than instruction by the artificial intelligent agent. Of course, the sample size is small, but with support from the National Science Foundation, we will carry out larger experiments soon, and expect them to follow the same pattern.

Intelligent Proof Mining

We confess to having dreamed of a system that would be capable of extracting a student's line of reasoning from electronically submitted homework assignments, and of providing a deep analysis of their work — and we suspect others harbor similar hopes. This dream has begun to take shape in the form of an agent that we affectionately call HILBERT. Our goal is to 'digitally capture' the reasoning process of students as they learn first-order deductive CIDR, and to have our intelligent agent provide student-relevant, cogent advice based on that process. Our method: Intelligent Proof Mining (IPM).

HILBERT - An Agent Architecture for IPM

Since our approach, as noted above, is agent-centric, our first inclination is to represent the problem of automated proof analysis in terms of a set of inputs (percepts), processes, and outputs (actions). As raw input, HILBERT (Figure 6) receives the student submission, and a set of *proof plans*, which represent a set of plausible solutions to the exercise that mesh well with the style of proof construction the human instructor wishes to teach the students. These inputs are pre-processed into *Deductive Dependency Graphs* (DDG's), and *InfoSets*. HILBERT receives queries from the proof plans, which seek to discover whether the student was able to correctly make the assumptions and deductions needed to solve the proof. Proofs are also composed of recurring patterns of reasoning. Many logicians, mathematicians, and computer scientists take for granted that they almost automatically apply certain transformations to formulae when they are engaged in complex reasoning. While a professional logician may intuitively know to use a tautology such as $\phi \vee \neg\phi$ in trying to prove a statement ψ , your average student in introductory logic would be completely stumped as to why the logician chose to use that particular construction, and how it is useful, given the circumstances.

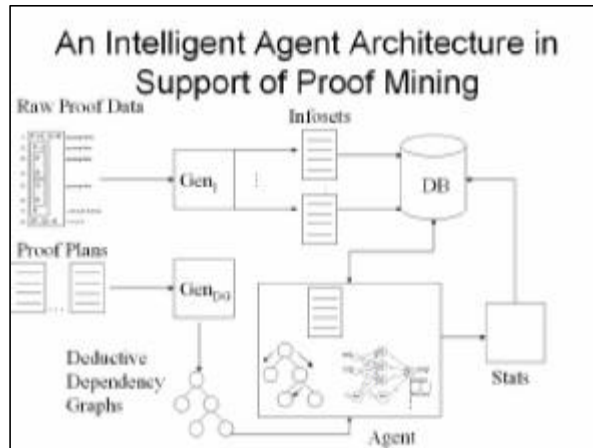


Figure 6. HILBERT: An IPM Agent

We are well aware of the need to teach students how to recognize and apply these constructions while engaging in a proof exercise. To account for this need, a copy of the student-submitted proof is read by HILBERT's Spatio-Temporal Pattern Recognizer: a committee of neural networks that identifies commonly used inference patterns, and makes judgments about the complexity of a given step in the exercise. This information is analyzed and processed into a set of real-valued probabilities that represent HILBERT's beliefs about the student's current state of knowledge, conditioned on the complexity of each step in the analyzed proof. By using conditioned metrics, students don't get penalized as harshly for mistakes nested in deeper levels of the proof. We will briefly discuss each highlighted component, and fit them into an overall architecture for mining repositories of natural deduction proofs. Techniques for automatic proof analysis are still in their infancy, with the first major developments coming out of Stanford's CSLI group in the form of *The Grade Grinder* (Barwise, Etchemendy 1999), which automatically checks the correctness of rules of inferences in proofs submitted over the internet. To ease exposition, we choose to focus only on the propositional calculus, which is a proper subset of FOL. The standard five truth functional connectives are accompanied by the special symbol \perp representing a contradiction. But first, we shall take a moment to explain the framework within which we are to perform our analysis.

Natural Deduction and Fitch-Style Proofs

We prefer the Fitch style of natural deduction to specify proofs. Illustrated below in Figure 7 is a proof solved as a Fitch-style natural deduction proof that we'll refer to as P^h :

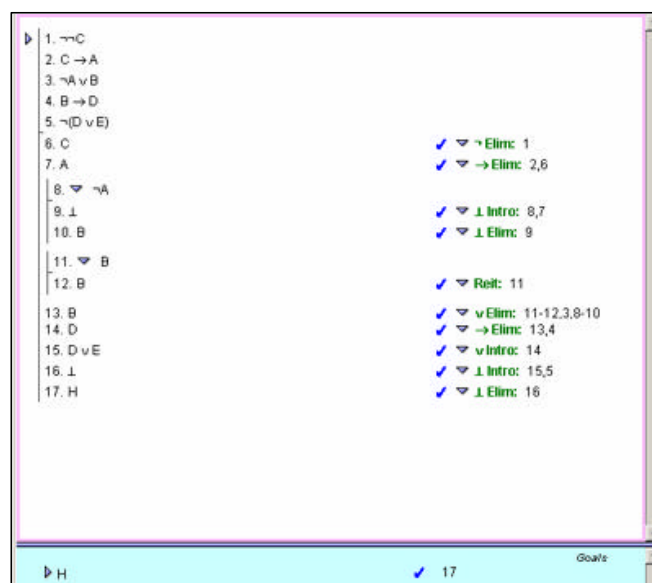


Figure 7. Proof P^h

There are several important features to note about this style of developing proofs. You can see that the proof is modular, in terms of ‘levels’ or nesting. These nested ‘subproofs’ (as we’ll be referring to them hereafter), are suppositions used to move deductively from the premises to the conclusions. You’ll also notice that the proof is organized into four distinct columns, each representing the line number, the formula, justification, and rule application respectively. This information is crucial to the mechanized analysis of proof information. This particular representation is visually helpful as well, as premises and assumptions are easily distinguished from deductions by the ‘Fitch Bars,’ which separate them.

Proof Plans, DDG's, and InfoSets

A Proof Plan is a high-level description of requirements (in terms of subgoals and inferences) for completing a proof. There can be many proof plans that correspond to a single proof, as there are an arbitrary number of solutions to a given exercise. These proof plans are either manually generated by instructors or automatically generated using an automated theorem prover plus some heuristics for converting output into Fitch-style natural deduction format. Some examples of what you’d find in a proof plan are the following:

Subgoal b - Requires: b as Assumption \perp as Subgoal, $\neg a$ as Assumption: b as Subgoal.

Deduce b Using Rule: \vee Elim Using SOS $\{b: b, \neg a: b, a\}$.

Briefly, the first statement says that in this given proof, there is a requirement to recognize b as a subgoal, and understand that to deduce b ; one must use two assumptions (b , and $\neg a$), and conclude that b follows in both cases. The second statement checks to see if the student used the proper inference schema and cited the appropriate justification steps (each subproof along with the premise $\neg a \vee b$ in this case).

Proof plans define the functional dependencies between formulae, assumptions, premises, and conclusions by explicitly enumerating subgoal requirements. We’ve developed DDG’s as an intuitive, visual depiction of these requirements that also happen to have some very desirable properties that HILBERT may exploit in its analysis. As you see below in Figure 8 the DDG is a directed, acyclic graph with nodes (vertices) containing information about each line of the proof, including line number, nesting depth, justification (in the form of its edges), and rule application (not shown here).

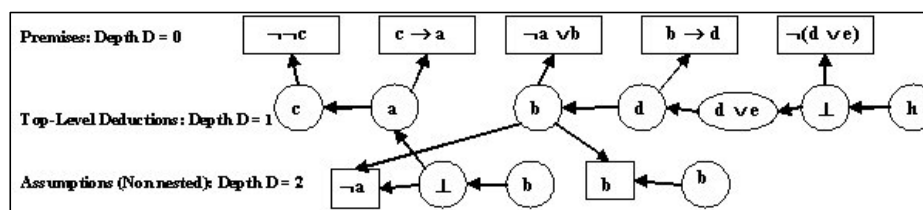


Figure 8. DDG for P^h

DDG's are a visual representation of the interactions between formulae, premises, goals, and assumptions in a proof. The in-degree of each node in the graph represents the number of other statements that depend on the node in question. The out-degree of each node encodes the number of justification steps required to validly claim the formula which the node contains. As you can see in the figure above, assumptions and premises (shown in boxes) have an out-degree equal to zero, whereas inferences (shown in circles/ovals) always have an out-degree greater than zero. DDG's can easily be drawn in ‘layers’ as shown above, to visualize the naturally layered structure of the proof, with each degree of subproof nesting on it’s own layer. When the intelligent agent first encounters proof data submitted by students, it will be in a raw form that is unable to be processed analytically. We use the layered structure of the DDG to define InfoSets, which are used to formally represent the notion of scope in natural deduction.

An InfoSet of order 0 (I_0) in proof P^h is defined by those formulae that comprise the given information in P^h . An InfoSet of order 1 (I_1) is defined by the union of all *connected components* (Corman, Leiserson, et al. 2001) in the DDG of P^h having nodes of depth 1. An InfoSet of order $1 + N$ is the union of I_0 , I_1 , and each connected

component of the DDG existing at a level less than or equal to level N . InfoSets in the figure above would thus be listed as:

$$I_0 = \{c, c \rightarrow a, (\neg a \vee b), b \rightarrow d, \neg(d \vee e)\}$$

$$I_1 = \{c, a, b, d, (d \vee e), \perp, h\}$$

$$I_{2a} = \{a, b\}$$

$$I_{2b} = \{b, b\}$$

As you can see, the last two InfoSets naturally encapsulate both assumptions in the proof. The InfoSet structure has a natural ordering (since the nodes that comprise it contain the line number in the proof at which they appear), and allows us to define a Set-of-Support (SOS) at each line of the proof. The SOS contains those formulae that are reachable from a given node (i.e., able to be used as legal justification in the inference process) in the DDG, plus all formulae in the set I_0 . For a quick exercise, find the node on the DDG that corresponds to the statement of contradiction at depth 2. Traveling along the outbound directed edges of the node, we can see that all reachable statements plus the set of premises gives us the set:

$$\{a, \neg a, c, \neg \neg c, c \rightarrow a, (\neg a \vee b), b \rightarrow d, \neg(d \vee e)\}$$

This set is the SOS. The SOS provides us with valuable information as to what a student was able to do at each step in the proof, given what he had deduced at any given point in the process. Breaking up a proof into InfoSets is a natural way to enable the agent the ability to make queries at different levels of abstraction. For example, consider these two queries:

Did the student recognize the need to perform \rightarrow Elim to arrive at “a” before she was able to deduce “b” using \vee Elim?

Did the student correctly identify the inference-schema to deduce “b” and did she cite the proper formulae as justification?

Each of these queries reduces to basic operations that can be performed on DDG’s, proof plans, and InfoSets. The first query asks to find a node corresponding to b on the DDG, making sure that the applied rule at that node is \vee Elim (if not, throw an error flag), and then requests that the SOS be checked at node b to see if a is a member. If it is, our question has been answered. For the second query, use the DEDUCE statement in the proof plan that corresponds to node b , and check its outbound edges, or justification steps, to see if they match those listed in the proof plan. While these queries provide a high degree of linguistic/symbolic expressiveness, we still haven’t addressed the problem of complexity, and how to model short-term memory, or the limited ability of the untrained mind to focus on many interrelated tasks at once.

Spatio-Temporality and Proof Structure

The correct identification and usage of inference patterns are two of the most crucial skills to acquire when learning to construct proofs. These skills have been largely, if not totally, ignored as performance metrics when building automated proof-analysis tools. Looking for inference patterns in proofs presents a set of very interesting challenges that range from the purely philosophical to the practical. For our first experiments, we’ve chosen a set of common patterns of inference that HILBERT will search for in a submitted proof. We treat the proof as a spatio-temporal structure, where a pattern is defined as a sequence of deductions (the temporal dimension), appearing to include commonplace term/operator/rule/justification orderings (the spatial dimension). For example, let’s consider *Hypothetical Syllogism*: To deduce a statement of the form $A \rightarrow C$ in this manner, the formulae $A \rightarrow \mathbf{f}$ and $\mathbf{f} \rightarrow C$ must appear before the deduction of $A \rightarrow C$ in terms of where they are located in the proof. To have a network be able to declare that the student has correctly used *Hypothetical Syllogism*, it must be able to locate the statements $A \rightarrow \mathbf{f}$ and $\mathbf{f} \rightarrow C$ in the presence of noise (other deductions, subproofs, and inference patterns) based only on their structural form and position with respect to the final deduction $A \rightarrow C$.

Recurrent Neural Networks

Elman's recurrent neural network model (Elman 1990) is one of the simplest, yet most elegant, spatio-temporal processing systems. The general intuition is that the system state which resulted from the last presentation (at time $t - 1$) of the input is looped back as an input to the network at time t and provides us with a contextual cue with respect to the current input that is being presented to the network.

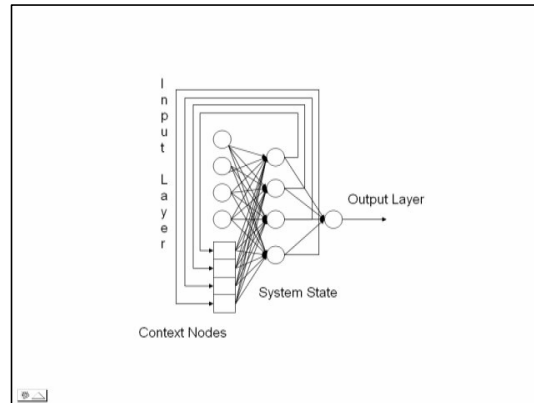


Figure 9. Elman Network

Elman's model is best known for its prominent use as a tool for connectionist analysis of natural language. It has seen remarkable success as a predictive utility; this success has to some degree inspired the current focus of our research. In (Elman 1990), the Elman recurrent network was used successfully to predict when words appeared in a long sequence of letters by associating letter sequences of different length and commonality. Our problem of detecting inference patterns in proofs is complicated by the fact that patterns have no limits on locality. We are faced with a conundrum: there may be an overall pattern that defines the whole proof, while other inferences that are used to achieve the necessary intermediate subgoals which define the overall pattern may be considered 'noise'. It has become clear that a single network doesn't have the requisite power to classify at all of these different levels of resolution, and at the same time have the extreme tolerance to noise that is required of our problem. We turn to biology to provide us with the answer to our dilemma.

Biologically Plausible Neurons

An interesting approach to noise-tolerance has been proposed in (Arbib 1997), taking the form of a biologically plausible neural processing element called the 'Leaky Integrate-and-Fire Neuron' (LIF for short). LIF's, like many other neural processing elements, have an activation associated with them, which defines the neural response to seeing a particular set of inputs. LIF's have the desirable property of continuous activation through time. When excitation of a LIF occurs, the activation begins a process of decay known as 'leakage,' whereby the activation value of the neuron loses some of its potential. So in essence, for a pattern to be recognized correctly, the constituents need to be seen by the network in the proper order. The potential of the neuron at time t is given in the equation below:

$$\tau \frac{dm(t)}{dt} = -m(t) + \sum_i w_i X_i(t) + h$$

Equation 1

where τ is the time constant, $m(t)$ is the potential, $X_i(t)$ is the firing rate of the neuron at the i^{th} input, h is a unique equilibrium point where the neuron is essentially neutral in the computation, and the w_i 's represent synapse strengths.

Each portion of the overall pattern excites the neurons in the network a little further, with the neurons ‘integrating’, or summing, each new correctly ordered constituent pattern till a threshold value is reached and the overall pattern has been fully observed by the network.

HILBERT’s Spatio-Temporal Pattern Recognizer

Since many of the patterns that we wish to classify are sets of statements that have similar structure (syntactically speaking) and temporal ordering, we appeal to a committee of classifiers (Figure 10), each encoding for a different inference pattern. It has been demonstrated (Jacobs 1991) that modularity may increase network training speed and efficiency. HILBERT is equipped with a set of distributed recurrent leaky-integrator neural networks; each network acting as a classifier for a specific inference pattern. These networks are connected to one another so as to facilitate lateral inhibition in the presence of ambiguous input patterns. More clearly, if an input pattern activates several of the classifiers upon presentation to the committee, ties will be broken as more and more patterns are shown to the networks and evidence is built up toward a high-confidence hypothesis. At each time interval (represented by the number of steps in the proof being presented to the committee), a set of observations is output by the network.

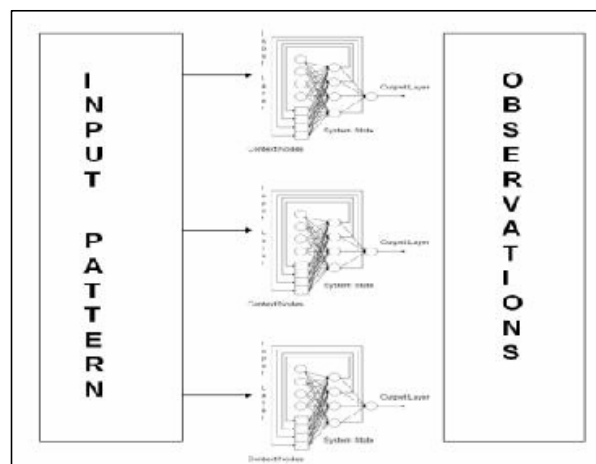


Figure 10. Committee of Experts

Cognitive Overloading

It has been argued that the most prominent limiting factor in learning is the capacity of our working memory (WM) (Anderson 1983). Many cognitive psychologists seem to agree that working memory is only capable of storing a very limited number of ‘chunks’ of data before forgetting begins to take place. (There is of course the famous heuristic that only 7+/- elements can be held in WM at one time, going back to the psychologist George Miller 1956). This physical limitation is further confounded by the fact that CIDR (at a non-trivial level) is essentially impossible without a divide-and-conquer approach to setting intermediate reasoning goals with respect to the overall goal of the exercise. These goals are highly interrelated, and often can have complex ordinal dependencies. While our discussion seems to point toward grim prospects for those wanting to master CIDR, hope still remains. Experiments performed by (Doane 1999) seem to indicate that WM has less effect on task performance than previously predicted. These results lead us to believe that the *organization* of WM allows for mastery in domains that require the mind to remain focused on many different variables at once.

Deep versus Shallow Evaluation of Performance

It is common for most ITS’s to use real-valued beliefs to represent the student’s knowledge. To our knowledge, problem complexity has never been a conditioning factor in the formation of those beliefs. The nature of the complexity estimates given in the last section are empirically derived and normalized into a framework that represents measured cognitive load. The ultimate goal of this work is to provide a capability to gauge learning by directly measuring change in working memory organization. If students begin to perform at non-novice levels in

the presence of potential cognitive overload, we can make a substantiated claim about true learning gain, which may very well be unprecedented.

Fused Metrics for Student Modeling

Trying to get a grip on what exactly constitutes knowledge of an inferential schema is a difficult task. Our experience tells us that performance can be measured as functions of whether the student:

- Correctly identified the need to use a specific schema in context of the given information.
- Used the schema correctly in context of the available information.

Bayesian Formulation

We will refer to the probability of a student knowing an inferential schema as K_i , and the probability of observing a certain set of contextual information as C . Let K_i be the joint distribution consisting of the probability that a student correctly identifies the need to use a pattern in the correct context (labeled as x), and that the pattern is used correctly in that context (labeled as y). K_i represents the probability that a rule or pattern (labeled with the subscript I) is in the known state.

$$\sum_{j=1}^n \sum_{k=1}^m p(x_j, y_k)$$

Equation 2

The intelligent agent maintains a set of beliefs about the student in the form of an array of non-negative real numbers, scaled between 0 and 1. Each inference rule and reasoning pattern is represented as a belief about the student inside of the agent. These beliefs represent the probability that a rule or pattern is in the 'known' state by the student. We use a traditional Bayesian formalism for maintaining and updating these values.

$$P(K_I | C) = \frac{P(C | K_I)P(K_I)}{\sum_i P(C_i)}$$

Equation 3

The above equation defines the conditional probability that a rule is in the known state given the context in which it was used. Information about context will be generated as a normalized weighted sum of the results of spatio-temporal pattern recognition and traversal of the DDG structure.

Toward a MetaLogic for Critical Analysis

It is our intention to extend the software implementation of the architecture in the coming months as part of an ongoing effort to produce a third-generation (Conati, Van Lehn 1999) intelligent tutoring system for introductory logic. While HILBERT v1 is capable of identifying correctness and complexity, HILBERT v2 (under development) will have the capacity for formal meta-reasoning using higher order logical systems. Even as we are currently able to answer questions such as "Did the student apply this rule correctly?", HILBERT v2 will be able to answer the more difficult question "Why was the student unable to prove this assertion?", and provide formal justification for the conclusions that it draws. Our enhancements as we move forward will in large part be based on mental metalogic, a new theory of human and machine reasoning created by Yang and Bringsjord (forthcoming).

The Complete Educational Package

Usage of information culled from student submissions within the broader scope of a real-time, agent-driven tutoring environment draws direct parallels to the 'hands-on' studio lectures that have become popular, while maintaining the deliberative, detailed analysis a student would receive on a graded test. Shown below in Figure 11 is a sketch of the interaction between our efforts in advanced theorem proving, real-time interactive agents, and intelligent proof mining technologies:

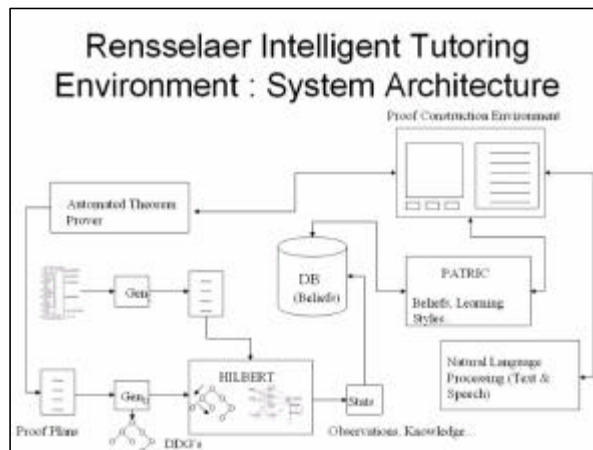


Figure 11. RITE (Rensselaer Intelligent Tutoring Environment)

In effect, HILBERT coupled with the agent-driven, interactive ITS technology (Bello, Bringsjord 2002) such as PATRIC being developed in the Rensselaer AI & Reasoning Laboratory will constitute a total educational package, fusing the best of lecture/recitation, studio exercises, office hours, and a full-time, personalized tutor that never tires of teaching.

References

- Anderson, J. R. (1983). A spreading activation theory of memory. *Journal of Verbal Learning and Verbal Behavior*, 22, 261-295.
- Anderson, J. R., Corbett, A., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: lessons learned. *Journal of Learning Sciences*, 4, 167-207.
- Arbib, M. A. (1997). Leaky integrator neuron. In E. Fiesler & R. Beale (Eds.), *Handbook of Neural Computation*, B1.4, Philadelphia: IOP Publishing and Oxford University Press, retrieved July 30, 2003 from http://www.iop.org/Books/CIL/HNC/pdf/NCB1_4.PDF.
- Barwise, J., & Etchemendy, J. (1999). *Language, Proof and Logic*, New York, NY: Seven Bridges.
- Bello, P., & Bringsjord, S. (2002). Agent-based real-time pedagogy for proof construction. *Paper presented at the Computing and Philosophy Conference*, August 8-10, 2002, Carnegie Mellon University, USA.
- Cohen, J., Kulik, J., & Kulik, C. (1982). Educational outcomes of tutoring. *American Educational Research Journal*, 19, 237-248.
- Conati, C., & VanLehn, K. (1999). Teaching meta-cognitive skills: Implementation and evaluation of a tutoring system to guide self-explanation while learning from examples. *Paper presented at the 9th World Conference on Artificial Intelligence and Education*, July 18-19, 1999, Le mans, France.
- Corman, T., Leiserson, C., Rivest, R., & Stein, C. (2001). *Introduction to Algorithms* (2nd Ed.), Cambridge, MA: MIT Press.

Doane, S., Sohn, Y., & Schreiber, B. (1999). The role of processing strategies in the acquisition and transfer of a cognitive skill. *Journal of Experimental Psychology: Human Perception and Performance*, 25, 1390-1410.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14, 179-211.

Jacobs, R. A., Jordan, M. I., & Barto, A. G. (1991). Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks. *Cognitive Science*, 15, 219-250.

Miller, G. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review*, 63, 81-97.

Russell, S., & Norvig, P. (1994). *Artificial Intelligence: A Modern Approach*, Saddle River, NJ: Prentice-Hall.

Sheines, R., & Seig, W. (1990). Computer environments for proof construction. *Interactive Learning Environments*, 4 (2), 645-665.

Suppes, P. (1981). *University-level computer-assisted instruction at Stanford 1963-1980*, Stanford, CA: IMSSS.

Yang, Y., & Bringsjord, S. (forthcoming). *Mental MetaLogic: A New, Unifying Theory of Human and Machine Reasoning*, Mahwah, NJ: Lawrence Erlbaum.