An Inquiry-based Approach to E-learning: The CHAT Digital Learning Environment

Steven Weisler, Roger Bellin, Lee Spector, and Neil Stillings

***Abstract*--This paper describes CHAT, educational software intended for use in introductory college-level syntax or general linguistics courses. The software is motivated by two persistent pedagogical problems frequently encountered in scientific teaching: student motivation and the abstract, non-procedural character of formal theories. After describing the system's agent-based architecture and interface, we argue that inquiry-based, collaborative software best supports the teaching of linguistic theory, and consider some broader implications for general science teaching in an e-learning environment.**

***Index terms*: collaborative learning, educational software, inquiry-based learning, science education**

I. Teaching Linguistics as a Science

Although the extent to which Chomskyan linguistics is a science (on a par with natural sciences, like physics, for example) has been debated on philosophical grounds [3], it is clear that the teaching of linguistics enjoys many of the same challenges as do its more traditional sister disciplines. For one thing, learning introductory linguistic theory means learning grammars–formal systems of rules and principles that determine in an algorithmic fashion the patterns of the languages of investigation. This is painstaking work both at the theoretical level (at which putative universal generalizations must be tested) and at the descriptive level (at which often-complicated sets of data must be analyzed). As a second challenge, students are often surprised to discover what the work of theoretical linguists is actually like. Seduced by questions about ape language and the thought-influencing lens of linguistic determinism, students are, to say the least, not eager to accommodate to a regime of data collection, hypothesis testing, and systematic analysis. Moreover, grammars are formal systems that depend on abstract notation and require analytic precision. From many students' point of view, learning grammars feels more like learning math and science than like the excursion into the humanities they were expecting.

In essence, the successful teaching of introductory linguistic theory requires solving many of the standard pedagogical problems confronting general science teaching, complicated by the student's pervasive sense that the field turns out not to be "what I thought it was." This paper describes the conception and development of CHAT, educational software that combines an inquiry-driven pedagogical framework with elements of proceduralized design in a collaborative digital learning environment. The goal of this project is to develop and assess new technologically supported teaching paradigms for inquiry-based education in the sciences.

II. Student Motivation

The introductory syntax course is central to the undergraduate linguistics curriculum but it is a difficult course for many students. The problems stem in part from the course's heavy (and probably unexpected) use of formal mathematical notation and reasoning. Some degree of mastery of formal syntax is necessary for students to appreciate the many spectacular results of modern linguistics, and because few entering students have such mastery, they typically begin with little motivation to shoulder the mathematical burdens of the course. The maintenance of student motivation is therefore an important challenge for syntax instructors. To approach this challenge, CHAT develops three leading ideas: it provides technical scaffolding for student inquiry, it promotes a collaborative learning environment, and it

---

Submitted to Scuola Superiore G. Reiss Romoli (SSGRR) 30.May.2001. The authors are faculty and staff of Hampshire College, Amherst Massachusetts. All correspondence should be directed to Steven Weisler, School of Cognitive Science, Hampshire College, Amherst Massachusetts 01002 (email: sweisler@hampshire.edu). This work was funded in part by NSF grant 8-0-0-33901. This effort was sponsored by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30502-00-2-0611. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency (DARPA), the Air Force Research Laboratory, or the U.S. Government.

develops an intuitive, procedurally implemented approach for teaching linguistic theory.

The use of technology to enhance student motivation has a long history and a rich literature. One current trend is to provide technological supports for students engaged in self-motivated, inquiry-oriented learning experiences. This approach is reflected, for example, in a recent special section in the *Communications of the ACM* in which the authors write that "Learner-centered, problem-driven approaches to education... are most effective in engagement, motivation, and, through their problem-driven format, in providing a solid conceptual understanding" [8]. This practice is also consonant with the general pedagogical environment of Hampshire College, where CHAT was developed; Hampshire College has a long-standing and explicitly articulated commitment to inquiry-oriented undergraduate education [7]. We therefore took it as our goal to develop a tool that enhances the motivation and learning of beginning syntax students by providing technical support for student inquiry.

A second insight developed in CHAT is that motivation for formal problem solving can often be strengthened when groups of differentially talented students work in a collaborative learning environment. To this end, CHAT employs a chat room-like design in which students use the grammars they develop to "chat." This not only allows for a channel of communication, but also permits students to examine the work of others and to test their own hypotheses in a cooperative manner. To a large degree, this works against the essentially logico-mathematical character of linguistic theories, anchoring the pursuit of this knowledge in richer social transactions.

Finally, grammars are normally thought of as theories of a speaker's abstract grasp of a language–specimens of linguistic knowledge that underlie the common ability to produce or understand a word or sentence. It is frequently difficult for students to comprehend the precise ontological status of these so-called "competence theories" [3], which are easily confused with the parsing systems and systems of natural language understanding to which they are related, but typologically distinct. In CHAT, students learn to construct grammars by building a sentence generator. This has the joint effect of proceduralizing the task of grammar construction from the outset while also rendering the implementation relationship between the generator and the grammar more transparent. Put more simply, students can directly grasp the task of making a system that can "chat," which in turn promotes an intuitive sense of the role of grammars in sentence production that is nearly impossible to achieve by theoretical description alone.

Several previous projects have provided students with general technological supports for the "inquiry cycle," for example by providing "lab book" environments integrated with statistical packages and graphing tools. The goal in the present project was to provide more direct and domain-specific support for linguistic inquiry by allowing students to explicitly construct and test linguistic theories in the form of grammars and lexicons. The "linguistic theories" in CHAT are active computational systems that produce behavior and data that students can observe to validate or falsify their hypotheses about the nature of language. This use of computational "system construction" (or "simulation construction," "model construction," etc.) as an active form of theory construction for inquiry-oriented education also has a history in the literature (see, for example, [10]), but to our knowledge CHAT is the first application of this idea to the field of linguistics. The linguistics domain is a particularly interesting application area for this approach since by using CHAT, students can typically understand a theory's strengths and shortcomings simply by examining the sentences it generates, using native ability as an English speaker to detect ungrammaticality.

III. The Construction of Grammars

A grammar consists of a lexicon (or dictionary) plus a set of context-free rewriting system augmented by a complex matrix of subcategory features that allow certain linguistic dependencies to be naturally captured. The lexicon contains a list of words along with information about syntactic distribution, part of speech, and other morphological properties (like singular/plural, masculine/feminine, mass/count, etc.), each of which must be properly established on pain of over- or under-generating a corpus of the target language. For example, the lexical entry for the "cat" would indicate that it is a third-person singular noun that requires a determiner (e.g., "the") as its "specifier"–that which introduces a common noun.

The phrase structure grammar takes the form of a set of rewriting rules:

S → NP VP
NP→ Det N
VP→ V Adv

In this simplified example, assuming a proper lexicon, the rule set suffices to generate the following syntactic analysis tree:

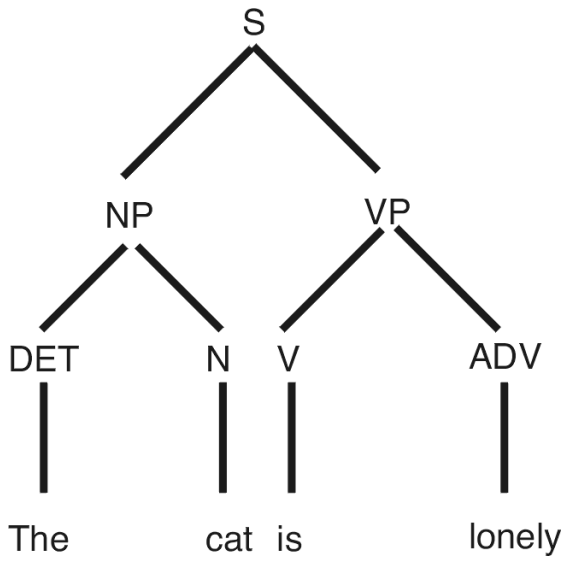

Fig. 1. A Syntactic Analysis Tree.

Reading the "→" as "consists of," the first rule sanctions sentences (S) that consist of a N(oun) P(hrase) followed by a V(erb) P(hrase). The categories "Det," "N," "V," and "Adv" stand for Determiner, Noun, Verb, and Adverb, respectively.

The central descriptive task is to expand the lexicon and the re-writing system to account for all and only the grammatical sentences of English (or any other language of description), a problem of great scientific magnitude. Ultimately, we confront aspects of syntax that require additional descriptive devices for example, the case system (e.g., the difference between "I" and "me"), the pronoun system, non-declarative sentence types, embedded sentences, and the helping verb system, to name just a few advanced topics in English syntax. CHAT, or indeed, any other learning environment for grammar construction, must scale up to permit the description of these more complicated data, as well the simpler phrase-structure dependencies exemplified by the grammar above.

More challenging still is the task of writing grammars that are independently motivated. Each proposed grammatical analysis makes a claim about the structural representations that a speaker assigns to a sentence. Therefore, many grammatical analyses that succeed in generating analysis trees for a range of target sentences nevertheless assign implausible structures that lose generalizations and ultimately fail to extend to provide a general account of the syntactic structure for the target language. Such tempting but incorrect alternative hypotheses must be rejected, often on the basis of further data. One of the most important heuristics that researchers must develop is a sense of where to look for independent support for a promising analysis. Often, the absence of such intuitions causes great difficulty for the non-vitiate linguist.

IV. Overview: The CHAT Concept

CHAT provides a networked "chat room" environment to which students and faculty can post sentences. The sentences, however, cannot be directly composed. Instead they are generated by clicking on a "generate" button that produces a parse tree for a particular sentence form, along with an interface that allows for selection of particular words for all of the tree's leaves. Only words that satisfy the constraints on the leaf and its position, as specified in the grammar and the lexicon, are made available for selection. When words for all of the leaves have been selected the sentence, along with its parse tree, is sent to the public chat room. The sentence generator is driven by a grammar, a lexicon, and several other parameters, all of which can be edited by the student using the graphical user interface described below.

Students typically begin with empty or primitive grammars and lexicons. Their goal is to make the system capable of generating sophisticated grammatical sentences (in some cases specific types of sentences suggested by the instructor), without allowing the system to generate ungrammatical sentences. In some cases a student with a problematic grammar may be able to disguise this fact by avoiding the selection of problematic words for particular tree leaves, but this trick can be easily exposed; sentences with *randomly* selected (but constraint-satisfying) words are also made available to others who wish to probe the grammatical analysis behind a particular sentence.

It is also possible to provide students with more complex initial grammars and lexicons, and to challenge the students to change/expand the system to account for new linguistic phenomena. CHAT's lexicon and grammatical rule-building tool kits are based on current Chomskyan minimalist syntactic theory [4]. The system allows for the investigation of a wide range of linguistic phenomena and in some cases allows for different theoretical approaches to

particular problems. As such, it is an open-ended environment for inquiry into the nature of human language. In this way CHAT's structured space for experimentation can provide a naturally incremental approach to theory-building. The student need not understanding all of syntax from the start, but can be asked to account for one syntactic phenomenon in the theory and then move on to another, progressively.

The network features of the system allow for collaboration among the students and faculty, and a community of software agents, described below, provides additional collaborative support and advice. The social nature of the chat room interactions, and of chat rooms in general, also facilitates student motivation and engagement. The software is intended for use in class or lab sessions, with 5-20 students (one or two students per computer) and with an instructor or TA present; but it could be used for individual experimentation or even homework assignments without networking. The chat room is based on TCP/IP networking, so it can be shared over the Internet among users around the world. CHAT is implemented in Java to facilitate distribution across all popular computer platforms.

V. Overview: The Agent Architecture

One challenge in the design of any software for inquiry-oriented education is how to provide useful feedback (help, advice, directions, etc.) without destroying the student-initiated character of the inquiry. This is difficult because the software cannot possibly predict all student actions or intentions when the inquiry is truly open-ended. To meet this challenge we followed another recent trend in the literature, that of using "intelligent agents" ([2], [1]) that function as learning companions ([5], [6]). These agents have more autonomy and intelligence than typical "online help" functions, but they do not attempt to fill the role of a teacher or tutor; each acts more like a "guide on the side" than a "sage on the stage" ([9]), providing advice only when consulted, signaling discreetly when new information is available.

To facilitate the development of robust agents we designed a general architecture for inquiry-based learning environments, in which students build and modify computational models (see Figure 2). The heavy arrows in the figure represent the normal interactive cycle of a software simulation environment: the user initiates actions that influence the simulation, and the simulation output is provided as feedback to the user. Our agent-based environment augments the simulation environment with two clusters of intelligent agents. One pool of agents watches the student's interactions with the system, along with a blackboard to which all agents can write, and reports significant patterns and trends to the blackboard. Some of these monitoring agents may attempt to model the student's understanding of the domain, but simpler agents that, for example, notice recurring cycles of values for simulation variables, will by themselves provide considerable utility. These "observer" agents can also access external information sources to obtain additional assessments of the status of the model. The second pool of agents, the "reporters," watches the blackboard and reports to the student. The reports may describe noted patterns, trends, suggestions for further experiments, and pointers to additional source materials. The reporting agents may also write to the blackboard so that future reporting agents can ensure that their reports are consistent and not redundant.

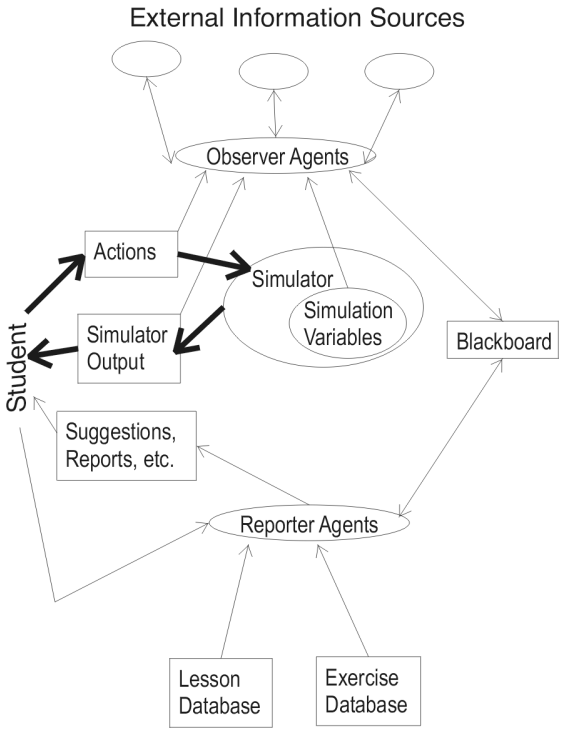

Fig. 2. Agent architecture for inquiry-based learning environments.

As a simple example of the general utility of this agent architecture, consider an inquiry-based tutor for epidemiology. The tutor might be built around a simulator that models the spread of diseases and immunity through a population. The simulator might include user-settable parameters such as disease transmissibility and average time from infection to death (for a fatal disease), along with dependent

variables such as the total number of infections. While a student is exploring and experimenting with the simulator, a simple monitoring agent might notice that, for example, the disease quickly died out in each of the student's simulations, and that the setting for time from infection to death was in each case unusually low. A reporting agent might then refer the student to source materials on the epidemiology of quick-killing diseases such as Ebola, while another might note other user-settable parameters that could break the noted pattern, and another might suggest experiments with specific settings of these parameters.

In many respects these monitoring and reporting features are similar in spirit to the monitoring and reporting features of simulation-based computer games such as SimCity and SimLife. These features give the user high-level status reports about the game/simulation as it progresses. This approach can be enhanced significantly through the use of good general representations, logical inference rules, and an agent architecture. When built around accurate simulators that are designed for educational rather than entertainment purposes, we believe that these features will provide substantial support for inquiry-based education.

VI. Overview: CHAT agents

The agents built for CHAT were motivated by observation of videotaped classroom sessions with an early (agentless) CHAT prototype. We noted ways in which students interacted with one another, and ways in which the instructor would intervene and guide students, and we attempted to build agents that act in similar ways. The architecture was designed specifically to ease the incremental addition of new agents, and we do not consider the current set of agents to be final.

The current implementation includes agents that *observe* (e.g., the "Link Parser Agent") and *report on* (the "Critic," the "Passed Rule Agent," and the Broken Derivation Reporter") a student's progress. The Link Parser Agent generates new sentences sanctioned by a student grammar "behind the scenes," and ships them off to the CMU online parser to check for unnoticed ungrammaticality. The Critic provides feedback to the students based on an analysis of the CMU feedback, while the Passed Rule Agent reports on phrase structure rules that could not be implemented because of a conflict in the relevant feature assignments. The Broken Derivation Reporter is responsible for reporting on circumstances in which an attempt to augment the grammar unwittingly results in giving up previously successful analyses of sentences that were generated by prior versions of the student's grammar. This reporter is particularly useful for students who are making large-scale changes to their grammars, especially in cases in which there are hidden implications of a change that are hard to deduce.

VII. CHAT's User Interface (Selected Features)

CHAT's user interface breaks the task of formalizing English syntax into logical parts, including program navigation, grammar and lexicon creation and revision, and windows for analysis tree displays, private, local generation of test sentences, a chat room, and the CHAT agents. At the beginning of each session with CHAT, a login dialogue is shown:

Fig. 3. The Login Dialogue.

The student may enter a name or nickname in the first field that will be used to identify the student's contributions in the chat room. The second field contains the Internet address of a computer that is running CHAT's server program. The address may be entered as an IP address (like "127.0.0.1") or a host name (like "host.college.edu"). If CHAT is unable to connect to this server because of an incorrect address or a network problem, or if the computer you enter is not running CHAT's server application, then the chat room will not function. (The student will be notified of CHAT's failure to connect, and all other parts of CHAT will continue to work normally.) Students using CHAT on a computer without network access, or not wanting to use the chat room to collaborate with other CHAT users can uncheck the checkbox labeled "Use chat room." In that case, no server address is required.

Students who have logged in can use CHAT's Navigation Palette to bring up the windows of CHAT

in which the various aspects of grammar construction may be undertaken.

Fig. 4. The Navigation Palette.

In particular, the Lexicon button makes the Lexicon window visible so that the student can create or edit a lexical entry, and the Rules button opens the Grammar window with which the student can create or edit a phrase structure rule. Here is a rather sophisticated grammar constructed in CHAT:

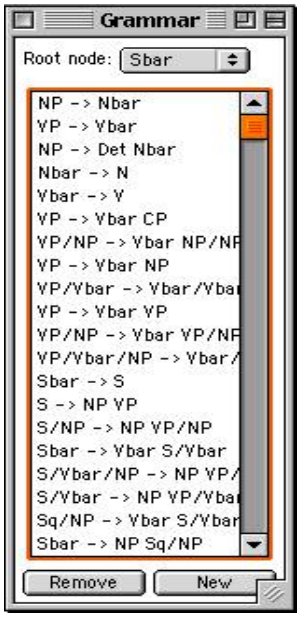

Fig. 5. The Grammar window.

Clicking the New button opens the Rule Editor window allowing the student to create new phrase structure rules in the grammar:

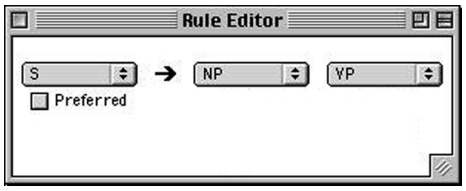

Fig. 6. The Rule Editor.

Clicking on any of the pull-down menus allows the student to select (or create) different node labels (e.g., S, NP, etc.). Clicking Preferred increases the probability that a given phrase structure rule will be used in generation to make it easier to test particular portions of the grammar.

Similarly, clicking the lexicon button pulls up the following window containing a list of previously entered lexical items:

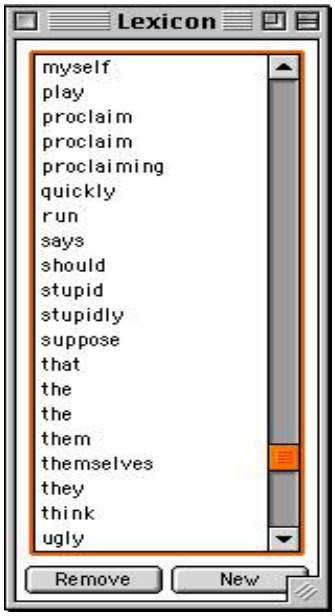

Fig. 7. The Lexicon window.

By clicking New, the Entry Editor window is displayed, allowing the creation and editing of lexical items via a complex series of pull-down menus that permit morphological feature assignment.

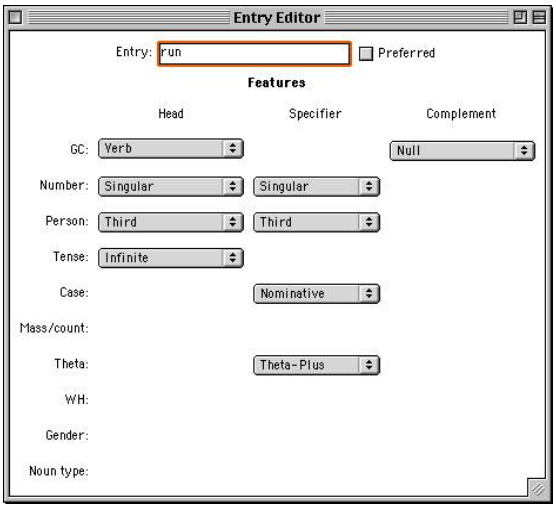

Fig. 8. The Entry Editor.

In a similar fashion, the Morphology button opens a window that allows the student to create and edit morphological rules (e.g., 'turn a singular noun into a plural noun by adding "s"'–general (but not exceptionless) rules that can be used to create new lexical items from old entries, thereby simplifying the construction of the lexicon. Finally, the Binding button plays a role in accounting for certain advanced properties of the pronoun system.

There are two options for students to test the output of their grammars. Clicking Generator opens the following private generation window:

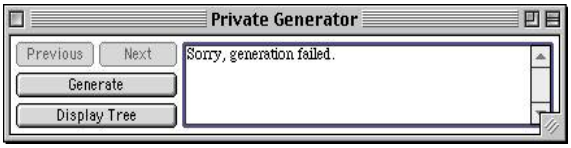

Fig. 9. The Private Generator.

In this mode, only the student (or a group of students working on a single machine) can see the output of the grammar. Every time Generate is clicked, a randomly generated sentence *compatible with the student grammar* is displayed in the generation field. By highlighting a generated sentence and clicking Display Tree, the analysis tree for that sentence, based on the rules and lexical entries contained in the student grammar, is displayed:

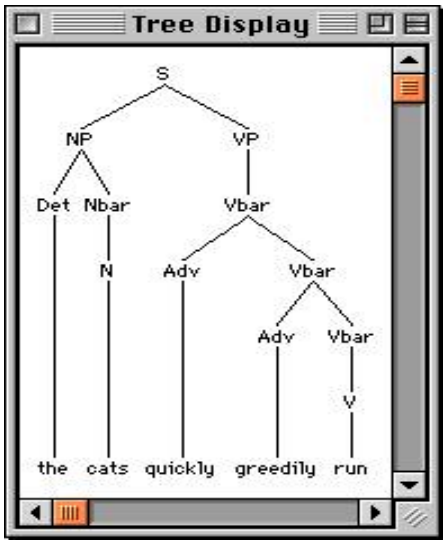

Fig. 10. The Tree Display window.

Alternatively, clicking the Chat button opens the Chat window through which the student can read the contents of and contribute to the chat room.

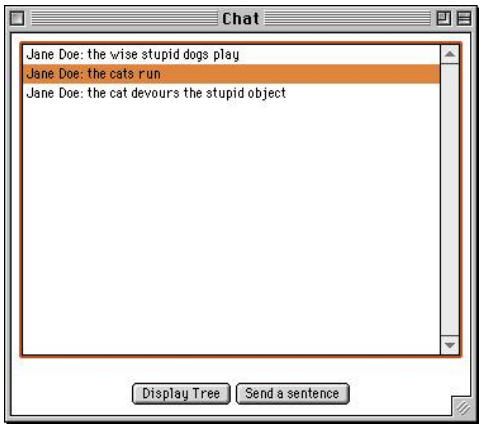

Fig. 11. Chat Room.

Clicking "Send a Sentence" generates new entries in the chat room. Each successive sentence generated (here, by Jane Doe) is listed sequentially in the Chat window along with its source, and is visible to all students running CHAT in a given session. By highlighting a sentence and clicking "Display Tree," an analysis tree for the selected sentence is displayed (see above), but in this case, the details for the tree for a given sentence are determined by the grammar that the *sender* used to generate it. Students not only witness the output of other students, but can also examine the details of the promising approaches of fellow students. In this way, CHAT promotes a truly collaborative learning environment.

The Agents button makes the Agents window visible so that the student can consult CHAT's agents:

Fig. 12. The Agents window.

The window displays a list of agents, each with indicators and controls below its name. Reporter agents prepare a helpful report to the student based on information gathered by the observer agents. The Status indicator light informs the student of the agent's status. If the agent is stopped, the indicator will be red; if the agent is running, the indicator will blink green. The Message indicator light is yellow if the agent has a new message to report since the last click of the Message button. The Details button creates an Agent Details window showing some under-the-hood details of the agent's activity, in case the student is curious or for debugging purposes. The Message button opens a Message window showing the message that the agent is reporting to the student.

VIII. Assessment

Two different earlier versions of CHAT have been used in the classroom, each with a different purpose in mind. The first deployment involved an agentless prototype of CHAT written in Director. Fifteen students in an advanced cognitive science class used the software in a section of the course devoted to linguistic theory. Two special evening sessions were reserved for this trial, which aimed at understanding how students would interact with CHAT (and at debugging the application). The two sessions were videotaped, and later coded, partly to evaluate the software and partly to better understand the inquiry process.

There were four main conclusions drawn from these sessions: 1) Student motivation for the task of writing grammars seemed considerably higher than that which is normally observed in traditional learning situations; 2) Students prefer to work in small groups of two or three at each terminal (and solo chatters seemed to make less progress); 3) It is important to provide a graded series of problems to guide students' inquiry; and 4) There were certain common kinds of feedback that should be provided by the software to strengthen the inquiry process.

The evidence for the improvement in motivation was, admittedly, anecdotal. Nevertheless, two separate evaluators were positively impressed by the degree of student involvement, the rate of progress through the assigned work, and the high-spirited atmosphere that pervaded the sessions. Students evidently enjoyed the exercise–a conclusion they bore out in post-test interviews. We were not surprised by students' desire to work collaboratively. However, whereas the chat room was explicitly designed to support group inquiry, we hadn't anticipated the students' tendency to work together at each terminal while using the "private" generator. Clearly, students gravitate towards a collaborative learning environment, and it is a positive design characteristic of the software that it promotes this style of learning.

The last two conclusions from the first sessions pointed to areas in which the software could be improved. Paradoxically, some very simple-seeming target sentences turn out to be tremendously complicated to analyze syntactically. Moreover, beginning students are not easily able to determine how difficult a particular construction is likely to be to generate with the resources of CHAT. Since it is demoralizing to run into serious descriptive roadblocks early in the inquiry cycle, we determined that students should be provided with a graded set of problems to support their progress in the preliminary stages of chatting. By carefully selecting the corpus that the student tries to generate, we can support a properly incremental learning curve.

Finally, we learned quite a bit about how the instructor can best interact with students while they

are using the software. Obviously, the role of the instructor is much different in the guise of CHAT facilitator than in that of lecturer. Although there is the occasional spontaneous formal lesson that is required when large numbers of students bog down, most of the teacher's time is spent moving from terminal to terminal, checking progress and offering suggestions. Since we are stressing the inquiry method, the instructor must suppress the urge to proffer solutions, attempting instead to steer the student down a profitable path. We took note of the fact that many students repeatedly required the same advice including help identifying hidden ungrammaticality and unextendable assumptions about structure. The desire to incorporate some of this scaffolding into CHAT lead to the development of the current agent-based model to monitor and provide systematic feedback on student work.

The second trial of CHAT employed a java-based beta version incorporating the aforementioned agent architecture. Twenty-two students in an introductory linguistics course used the software for four consecutive course sessions as their only course materials for the syntax module of the class. Work on CHAT was guided by a series of ordered problem sentences which students were instructed to try to program their systems to generate. At the beginning of each session, students could upload a starting grammar that incorporated the most successful parts of the previous session's work.

Student progress on required work was excellent. Most collaborative groups were able to successfully complete a series of open-ended questions that required applying lessons learned in CHAT to new problems. Since we have been using similar problems over the years in traditional syntax courses with only mixed results of achievement, we found the present level of performance encouraging.

Students were formally surveyed after each class to gauge their progress as well as their interest in using CHAT. Table 1 summarizes the results on seven student-response items that were included in the post-class assessments.

Item	Agree	Agree Somewhat	Dis-agree
1.The software allowed me to begin to actively explore how English syntax works.	16	6	0
2.I am learning more about syntax by using the software in class than I would listening to a good lecture in which the teacher also responded to student questions.	8	10	1
3.I could use the software to explore syntax outside of class, guided by appropriate assignments.	15	2	2
4.During the syntax classes so far, I have found it useful to work with (or consult with) a partner or neighbor while using the software.	11	2	1
5.My learning of syntax and enjoyment of the software has been aided by the CHAT feature.	12	2	0
6.Overall, using the CHAT software to learn syntax was a successful learning experience for me.	18	0	2
7.I find syntax more interesting than I thought I would.	14	4	1

Table 1: Student Response Items
Each row reports the number of students who gave each response to the item. Total number of responses vary because items were administered on different days.

Items 1 & 2 confirm the positive response of students to the active learning environment. In item 2 students appear to judge the learning outcomes in the CHAT environment and in a good lecture environment to be more similar than the instructor and evaluator believe they are. The difference may reflect students' overconfidence about what they learn from lectures. Item 3 suggests that CHAT should be evaluated as an out-of-class as well as in-class learning environment, although, again, students may be overconfident about how much progress they could make without an instructor present. Items 4 and 5 confirm students' beliefs that it is useful to collaborate, both directly with a classroom partner and via the CHAT feature. Items 6 and 7 suggest that CHAT is an effective and enjoyable learning environment overall. Item 7 was included because students typically report that syntax is the least enjoyable section of a linguistics course.

Two items were included in the assessments to check whether students' strong performance within the CHAT environment would show immediate transfer. The first item was: Draw a phrase-structure tree for the following sentence: *Healthy people exercise frequently*. This item required students to draw by hand the tree structures that are automatically generated by the software and to remember the basic structural generalizations they had learned in the first two classes without referring back to the grammars they had saved on the computer. Thirteen of nineteen students drew correct trees, and the remaining six students drew trees that were largely correct but contained one or two errors. In the instructor's and evaluator's past experience students have been unable to perform at this level in lecture-based courses. The second item was: Describe briefly something that you learned about syntax today. Sixteen of 22 students wrote answers that incorporated concepts that figured in their work during the class (e.g. I learned the same group of words can mean two different things based on how constituency is divided.) The remaining students commented on their learning process (e.g., I learned that syntax is hard) or wrote answers that did not mention specific concepts (e.g. I learned how to add additional words to sentences).

There was some evidence that working within CHAT led students to focus mainly on specific technical issues raised by the assignments. This possibility was suggested by responses to two items that invited them to reflect on their learning: (1) At this point what do you least understand or find most confusing about syntax?; (2) What aspect of syntax or issue about it would you most like to work on or have discussed in the final syntax classes? Nearly all answers focused on narrow issues that had come up in the day's work rather than on larger questions in linguistic theory. Lectures, readings, or more reflective assignments are needed to place the work within CHAT in a larger context.

Students were also given the opportunity to comment on things they liked best and least about the course. Many cited the textbook as a weak link and CHAT as the best part of the course (the fact that both were developed in part by this paper's senior author, notwithstanding!). Unfortunately, due to time constraints, neither the advanced grammatical features nor the agent architecture of CHAT were tested or evaluated in this test session. Finally, although this second evaluation is still partial (and a more systematic assessment must be forthcoming), we find that there is mounting evidence that CHAT can raise the levels of student motivation and accomplishment by successfully promoting an inquiry-based collaborative digital learning environment.

IX. Conclusions

The teaching of science to undergraduate students faces many well-known hurdles, among which are the apparent lack of applicability of scientific theories to the student's practical concerns, a tendency toward difficult formal methods and notations, the considerable abstractness of the problem domain, and the need for high levels of analytic rigor. Along with many other teaching scientists, we are committed to the development of inquiry-driven models of science instruction as a primary strategy for overcoming lapses in student motivation and for promulgating the capacity for scientific method that is most likely to sustain a life-long interest in the sciences [7]. In this paper we argue that educational software can play an important role in inquiry-based science teaching. By creating a digital learning environment that promotes collaborative work, CHAT manages to transform the normally unengaging process of grammar construction into a highly engaged, socially rooted learning experience. Perhaps more subtly, CHAT encourages precision by drawing on the student users' ability to evaluate their work (individually and collectively) and grounding that ability in a formal, computational system. It also monitors and reports back on the student's emerging work by employing a series of observer and reporter agents that provide useful feedback to the user.

More than just a static simulation, CHAT offers students the means to dynamically build a generative system that approximates their own native language ability. Ultimately, we are using a computational model to get across to students the conclusion that language is a computational system. We are confident that the transparency of this approach holds promise not only as the basis of an approach to teaching linguistics, but also as a generalized model for instruction across many scientific disciplines.

X. Acknowledgment

The development of CHAT was supported by grant 8-0-0-33901.from the National Science Foundation for "Inquiry-Based Learning: Cognitive Measures & Systems Support."

XI. References

G.A. Boy, "Software Agents for Cooperative Learning," in *Software Agents*, J. M. Bradshaw, Ed. Cambridge, Mass.: The AAAI Press/MIT Press, 1997.

J.M. Bradshaw, *Software Agents*, Cambridge, Mass.: The AAAI Press/MIT Press, 1997.

N. Chomsky, *Aspects of the Theory of Syntax*, Cambridge, Mass: MIT Press, 1965.

N. Chomsky, *The Minimalist Program*. Cambridge, MA: The MIT Press, 1995.

B.Goodman, A. Soller, F. Linton, and R. Gaimari, "Encouraging Student Reflection and Articulation using a Learning Companion," *The International Journal of Artificial Intelligence in Education*, Vol. 9, pp. 237-255, 1998.

P.Hietala and T. Niemirepo, "The Competence of Learning Companion Agents," *The International Journal of Artificial Intelligence in Education*, Vol. 9, pp. 178-192, 1998

A.P. McNeal and F.S. Weaver, "Interdisciplinary Education at Hampshire College: Bringing People Together Around Ideas," in *Reinventing Ourselves: Interdisciplinary Education, Collaborative Learning, and Experimentation in Higher Education*, L.S. Smith and J. McCann Eds. Boston, Mass.: Anker Publishing Company, 2001.

D.A. Norman and J.C. Spohrer, "Learner-Centered Education," *Communications of the ACM*, Vol. 39, No. 4, pp. 24-27, 1996.

M. Paolucci, D. Suthers, and A. Weiner, "Automated Advice-giving Strategies for Scientific Inquiry," in *Intelligent Tutoring Systems: Third International Conference ITS'96*, C Frasson, G Gauthier, and A Lesgold, Eds. Montreal, Canada, June *1996 [Lecture Notes in Computer Science*, New York: Springer, pp. 372--381, 1996].

A. Repenning, A. Ioannidou, and J. Phillips, "Collaborative Use and Design of Interactive Simulations. In *Computer Support for Collaborative Learning (CSCL),"* C. Hoadley and J. Roschelle, Eds. UNext.com. Press, 1999.

J.A. Self, "Bypassing the Intractable Problem of Student Modeling," *in Intelligent Tutoring Systems: At the Crossroads of Artificial Intelligence and Education.* Norwood, NJ: Ablex, pp. 107--123, 1990.

Steven Weisler, Ph.D is Dean of the School of Cognitive Science and Professor of Linguistics at Hampshire College in Amherst, Massachusetts. He is also a co-founder and the Director of the Innovative Instruction Laboratory, an interdisciplinary center for the conception and development of educational software that incorporates student-active, inquiry-based approaches to learning. Dr. Weisler's work in linguistics concerns the syntax-semantics interface, with supporting work in philosophy of language and linguistics pedagogy.