Integration-Kid: A Learning Companion System

Tak-Wai Chan

Institute of Computer Science and Electronic Engineering
National Central University
Chung-Li, TAIWAN 32054, R. O. C.
email: chan%ncu@twncu865.bitnet or ncut040@twnmoe10.bitnet
Fax: 886-3-4255830

Abstract

This paper describes a learning companion system called Integration-Kid in the domain of learning indefinite integration. A learning companion system is an intelligent tutoring system of a new breed. Apart from the teacher, a learning companion models after an additional agent, called the learning companion. The learning companion acts as a companion for the human student in learning. Thus the companion performs the learning task at about the same level as the student; and both the student and the companion exchange ideas while being presented the same material by the computer teacher. The computer companion might make mistakes, just like a human student.

1. Introduction

Intelligent tutoring systems (ITS), modeled after the idea of a private tutor, provide individualized instruction which offers the potential of better attention to individual student needs. This original vision remains strong today (Clancey, 1986). Realized as an interactive computer program, most ITS can be naturally conceived as a *two* agent model — a learning environment with a human student and a computer simulated teacher.

Learning companion system (LCS) (Chan & Baskin, 1988, 1990; Chan, 1989) is a *three* agent model (Figure 1). In this model, the computer simulates not one, but two coexisting independent agents — a tutor and a learning companion. The additional agent added to the traditional ITS model forms a small society with a richer social context. The simulated learning companion may act as a competitor or as a collaborator to the student. While being challenged by the companion, the student also observes sub- optimal performance of the companion. Such sub-optimal performance or misconceptions in learning are common, natural, and not limited to the human student.

The notion of a learning companion is related to collaborative partner and apprenticeship learning recently proposed by some ITS researchers. Self and his colleagues (Gilmore & Self, 1988; Cumming & Self, 1989; Dillenbourg & Self, 1991) suggest that a computer may be viewed as a collaborative partner to the student, able to offer advice and suggestions about the material in the learning process, instead of being a teacher. Also, a group of cognitive psychologists and ITS researchers (Brown et al., 1989; Collins et al., 1989; Newman, 1989; Brown, 1990) address that knowledge is "situated", being in part a product of the activity, context, and culture in which it is developed. In particular, they propose learning through apprenticeship-like methods. LCS is a model of ITS where learning is embedded in social activities of a particular simulated small society. Different from the collaborative partner and the apprenticeship model, the additional agent of the learning companion allows LCS to offer a richer social context as well as to provide competition and reflection in terms of a peer's sub-optimal performance, not just collaboration by the collaborative partner or scaffolding by the teacher. Common to these two models is the same emphasis on learning within a social context and the de-emphasis of the role of student model.

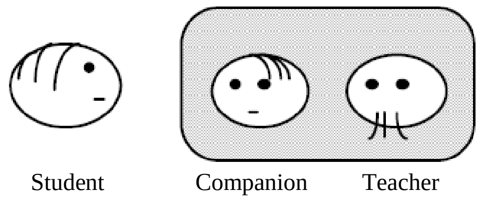


Figure 1 Learning Companion Model

The hypothesis of this research is: in the three agent model of learning companion system, more dimensions of learning may be achieved, higher motivation may be maintained, and better attitude of learning may be fostered, and thus, the three agent model is more educationally effective than the two agent model of traditional ITS. The contribution of this work is the establishment of an alternative architecture which enables realization of peer interaction and its implications into a computer-based learning environment. Integration-Kid has proved the feasibility of building such a system and we have noticed that it stimulates more dimensions for student's learning (e.g., collaboration and competition) than the traditional single-teacher oriented ITS environment.

Indefinite integration, at the level of a first year undergraduate, is the subject we chose to explore the design, construction, and operation of learning companion systems.

For example, $\int x \log x \, dx$, $\int e^x \sin x \, dx$, etc. Integration-Kid has been implemented with two man-year work using Common Lisp on TI Explorer.

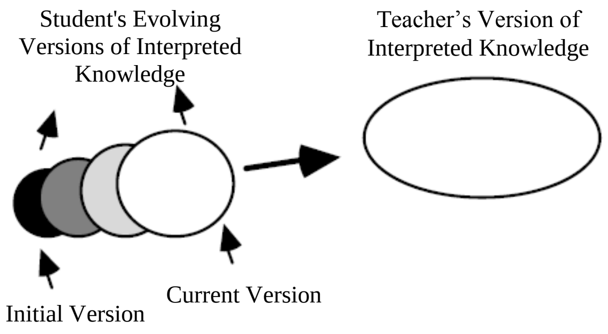
2. Psychological Perspectives of Peer Learning

Influence of social interactions on individual cognitive development has been observed by a number of researchers. First, peers can be sources of cognitive conflict. Piaget (1965) suggests that peer exchanges, especially those that bring different viewpoints into the child's awareness, are likely to play a role in the reduction of egocentrism. Second, peers can be providers of cognitive scaffolding. Scaffolding involves a kind of cooperative problem solving effort by both learners. Prominent among the viewpoints complementing Piaget's emphasis on disequilibrium is that of Vygotsky's hypothesis (1978) that social interactions play a fundamental role in shaping internal cognitive structures. Third, peers could be arousers of motivation which enhances a student's achievement-striving behavior. In particular, revealing strengths and weaknesses in comparison among peers can draw strong focus on students' attention and improvement. But such competition presents a number of ill effects (Collins et al., 1989). In Integration- Kid, with teacher's emphasis on the process rather than the outcome, students will learn that sub-optimal performance is inevitable and natural in the process of learning. Careful control of the performance of the learning companion and competence information to the student could avoid any decline of the student's intrinsic motivation but maintain its positive effects.

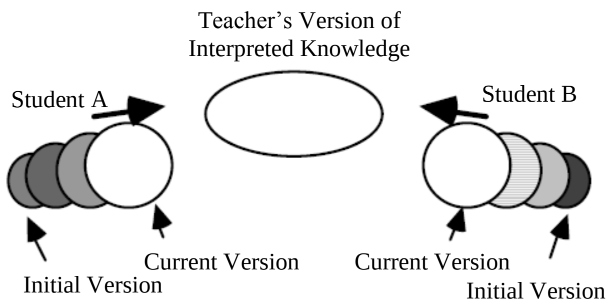
As an abstraction of understanding the "knowledge dynamics" in different learning environments, we may view the current knowledge of a person in a given domain as his own *interpretation* of the domain. This interpretation evolves from his previous interpretation as a result of learning. Such a change of interpretation involves his background knowledge, historic culture, and the current learning environment. To emphasize such individual dependencies, we call a person's knowledge of the domain his own interpreted knowledge. But, such interpreted knowledge of the domain varies over time, so, at a given point of time, we say that he has acquired a *version* of interpreted knowledge in the domain. Note that the interpreted knowledge may well consist of both correct and incorrect concepts. A version of interpreted knowledge is a *higher level* one than another if it articulates more knowledge, fewer misconceptions, and more flexible representation.

In the learning environment of a traditional ITS, the goal of the teacher is to try to alter the student's evolving interpreted knowledge so that it *converges* (Figure 2a) to his own (which is a much higher level but rather static one). An important function that the teacher exercises is monitoring of the learning activities and providing support for the student so that the convergence goes effectively.

In the situation of learning companion system environment, while one student is looking at multiple perspectives, planning, evaluating new ideas, monitoring and assessing solutions, he is forced to unfold, examine, and defend his ideas when challenged by the other student, and in turn to keep an eye out for possible mistakes made by the other which may result in defeating a proposed idea. Such a process of mutual justification may not easily happen in a student-teacher situation because of different social status and knowledge levels, and thus different expectations. Thus, learning, in an LCS environment, is the merging (Figure 2b) of two evolving versions of interpreted knowledge into the teacher's one.


Student's Evolving Versions of Interpreted Knowledge
Teacher's Version of Interpreted Knowledge
Initial Version
Current Version

(a) Traditional ITS Environment



(b) Learning Companion System Environment

Figure 2 Dynamic Cognitive Changes

From this view, *teaching* may be regarded as the effort of altering the other's version of interpreted knowledge towards his own while *learning* is working the other way round. In this regard, it is quite easy to see that in the student-teacher environment, one is teaching and the other is learning. But, in the student-student interaction, both agents are teaching and learning. Notice that, the level of a student's knowledge may never reach the level of the teacher's knowledge. For example, giving a problem of integral calculus to a student and to a professor of real analysis, even though they both solved the problem, we would never say that they have the same understanding of integral calculus. However, the presence of a teacher is necessary because of the imposing control on the student- student interaction; otherwise, their interpreted knowledge may not spontaneously converge to the knowledge defined by the learning goals. For young students, this is particularly necessary for they are not self conscious enough of their own learning.

3. Design Rationale

Before conceiving the conceptual design of LCS, we have conducted some tutoring activities with a small number of human subjects for learning introductory integration. Throughout the learning process, students seem to go through several *stages of learning* which we call the imitation stage, the developmental stage, and the integration stage. Moreover, we noticed that domain knowledge structure, learning goal management, and difficulty level of problems are related concerns to be addressed at all levels in the design of Integration-Kid.

We may tell the student that the learning companion is *simulated* to be his classmate, or, that the learning companion is actually *learning* together with him. These different sayings correspond to two different approaches of implementing the knowledge of the companion — simulating skill acquisition and actual machine learning. In the first approach, performance capability of the companion is controlled by the system while in the second approach, the companion is required to be able to *learn* as the student does but by using the techniques of machine learning (Gilmore & Self, 1988; Dillenbourg & Self, 1991). In both approaches, the image presented to the student must be that of a companion whose skill advances roughly in the same way as that of the student. While the machine learning approach may arouse the student's curiosity about how the companion learns and could possibly be one of the most interesting application areas of machine learning, the sophistication of the learning companion seems beyond the scope of current learning systems. In the simulation approach, the growing domain knowledge of the companion can be viewed as selective use of the knowledge preset by the system together with possible misconceptions of a typical student. In this work, we have adopted the simulation approach. In order to match a large population of students, the learning companion is simulated as a typical average student. Also, to be conceptually clear in modeling the complex learning interaction, we have decided that the companion to be modeled as an independent agent, not controlled by the teacher.

A declared role always represents a range of expectations to others. In LCS, the student will have different expectations from the teacher and the companion. We have designed several different *protocols of learning activities* in different stages of learning where the role of each agent is defined. For example, in practising basic integration rules, we have adopted competition format. But in solving miscellaneous problems, the protocol *responsibility sharing* decomposes the problem solving process into negotiation, decision, and execution for the student and the companion. One is responsible for decision making and the other is for execution of the decision made. The one who will make decisions first suggests a plausible strategy and explains. For example, he suggests using integration by parts and specifies what is *u* and what is *dv*, then he explains how the problem is similar to the previous one. Then the one who will work on the problem makes a different but plausible suggestion and explains why, if he finds one.

Next, the one who is responsible for making decisions decides which suggested strategy to use and the one who is responsible for execution works on the problem according to the decision. This procedure repeats until the next decision point. The roles alternate. The teacher interrupts if the learners get to nowhere.

Integration-Kid does not have a student model in the sense that there is no explicit representation of the student's knowledge. The purpose of student modeling in ITS is to induce a model of the student's knowledge from his observable behavior. Some researchers have recently questioned the need and the possibility of building such a student model (Self, 1988; Newman, 1989). In Integration- Kid, the companion's behavior can be viewed as a form of active student model. The companion uses a similar language to describe the problem, shares the same view and feeling, and shows sub-optimal performance in solving it which might have been a probable error by the student himself. Rather than used by the teacher, it interacts explicitly with the student and reflects to the student an image close to him.

Whether a student's response is buggy or correct, the student's response is either expected or unexpected to the system and has to be dealt with. In the early stages of learning in Integration-Kid, the student's expected buggy response is modeled as a part of the companion's behavior, e.g., applying a buggy rule. When the student makes such an error, in addition to the teacher's indication and explanation, the student can compare his bug with that of the companion's. In learning advanced methods where the student and the companion interact intensively, whether a learner's move is correct or buggy is the subject of their discussion. Thus, not only does the human student make buggy moves, the computer does also; not only is the computer responsible for discovering buggy moves, the human student is also. Learning under a rich social context such as Integration-Kid, the system can directly tell the student that his response cannot be understood, based on their past interaction and the current problem situation.

4. Architecture and Implementation

In designing the architecture to simulate the LCS interaction within a learning episode, it is natural to represent the three agents in the system separately. Each agent is a set of *rules of behavior* modeling the behavior of the agent. The three agents communicate via a *blackboard* through a simple *agent scheduler*. The blackboard is a collection of data representing the current situation. When an agent looks at the current situation, rules of behavior of that agent will be tested to see which one is appropriate to execute. Such rules of behavior are represented as production rules. Production rules exhibits some psychological validity in resembling how an agent looks and reacts to a situation. The protocols of activities are constructed with such rules of behavior.

Behavior of the human student is driven by his own intelligence. But, for the teacher and the companion, their general communicative behavior in learning is supported by their domain knowledge. Their problem solving abilities are modeled by the problem solver and are called by the action parts of the rules of behavior. This problem solving ability can be viewed as part of an agent's behavior. For the student agent, apart from those rules that interpret the student's input, there are rules that allow the student to control the system at his own pace such as referencing an old problem or the basic rules of integration.

The interface consists of four panes. The teacher, student, and companion panes represent the areas where each agent's utterances are displayed. The detail pane is for the student to reference some information.

4.1 Organization of Rules of Behavior

The LCS learning activities can be described by three levels of abstraction. The global level is the *curriculum level.* The curriculum is the whole discourse of learning activities in a certain structure which incorporates the domain knowledge structure and learning goals into the structure of learning activities. The second level is the *protocol level* which organizes learning activities in a certain format, as discussed above. The last level is the *episode level.* An episode is a basic unit of learning activities, which usually has a beginning and an end. In short, a curriculum consists of a set of protocols, a protocol is composed by a set of episodes, and an episode is a basic unit of learning activities.

Since the domain knowledge structure defines the learning goal hierarchy which, in turn, is the basis of the design of learning activities, we can represent the levels of abstraction by a *curriculum tree* structure (Chan, 1991a). According to this structure, the whole curriculum is the root node of the tree, the protocols are the internal nodes that are above the episode nodes, the leaf nodes. Rules of behavior of different agents which govern the learning activities are organized and distributed in this curriculum tree structure. That is, each node consists of a set of rules that are related to that part of the curriculum represented by the node. Two alternating phases are involved in running the system. In learning, that is, at a learning episode, rules of behavior of different agents related to that current episode node are invoked. After running the episode, then goes to the phase of scheduling. Scheduling rules which are resided in the internal nodes will determine which episode node to be visited next.

4.2 Modeling of Domain Knowledge of Companion and Teacher

The companion's problem solving behavior forms a basis of his interaction with other agents. First, the companion possesses certain background knowledge, such as simplification of algebraic expressions and differentiation. Presumably, he has no trouble with such skill. After the teacher introduces the basic concepts and demonstrates some simple examples, the companion acquires a set of basic rules of integration. These basic rules of integration are described as a form of *term rewriting rules.* Term rewriting rules are simple in syntax. Using a term rewriting system, we can offer a detailed explanation by showing for every step which rule to apply.

The set of rules is imperfect for there are incorrect and missing rules. Forgetting a rule means a rule in the rule base is missing and refining a rule means replacing the rule by another rule. The set of rules is fine tuned when the c$^{p\cdot}$o^{6}mpanion solves problems independently with the student and reveals problems with the rules. In Integration-Kid, there are 14 term rewriting rules for integration, 19 for differentiation, and 80 for simplification. It is a bit surprising that rules for simplification far out number the rules for the content to be learned, the rules of integration. Indeed, it takes years to learn the skill of simplification before one is ready to learn integration.

In learning new techniques such as the substitution method, the companion acquires procedures which incorporate the basic rules and his background knowledge such as differentiation. With scaffolding by the teacher, both the companion and the student have no trouble applying these procedures. However, successful use of such techniques hinges on the right choice of a sub-expression for the integrand, a choice which is largely heuristic in nature. Now, instead of generating and testing all possible candidate sub-expressions for each problem, the companion uses a set of plausible candidates as a base to hold conversation with the student via the protocol of activities.

For teacher, his problem solver is the same as that of the companion's, except at the outset, the set of rules of integration is complete and sound. Both the teacher and the companion adopt the same list of candidates and the situation map (discuss below) but their rules of behavior that define the protocol reason differently on it.

4.3 Situation Map — Representation of Problem Situations

In solving complex miscellaneous problems where almost every problem has its particular features that might not share with other problems, simply using rules of behavior to simulate such particularities may require a lot of rules for different problems as well as causing complexity in the action parts of the rules. Such a difficulty seems to be an insufficiency of representing the relationship among changing problem situations by using simple data in the working memory. Another concern of working on complex problems is the addition of social context to the current situation, for example, making a comment like "We are getting close." Such social context is usually situation specific to a particular problem. Moreover, personal subjective feeling, speculation, or even inspiration could possibly contribute to such social texture. Neither encoding such context within the rules of behavior which simulate the protocol of activities for a general set of problems nor using a simple data structure will suffice to represent such particular knowledge.

An explicit representation of problem situations which we call the situation map (Chan, 1991b) is used to alleviates the difficulty of representing complex problem situations. A situation map is essentially a graph data structure. Each node represents a problem solving situation and is filled with details of the particulars of that situation. A link relates two problem situations, and thus their previous interaction.

The companion takes situation map as the blue print to choose an action for different situations. The companion will treat this representation as his expectation to react to the student's alternative suggested action. When the companion is to execute, he will compare the student's decision against this representation. Thus, with a situation map, rules of behavior of the companion in the protocol become an interpreter of the situations represented by it during negotiation.

The use of a situation map have illustrated some characteristics. First, a situation map explicitly represents in an abstract way the situations in solving a particular problem so that it is general enough for different problems. As a result, the rules of behavior in an episode are significantly simplified. Second, it captures many possible situations, particular details including social texture, the corresponding possible actions, as well as how all these situations relate to each other. Thus, it is likely that for a particular student, the companion will only utilize partial information of this representation. Finally, in running an episode, the history of the past situations of solving the problem are recorded in the representation. Thus, during negotiation, the companion's decision reflects their past interaction.

5. Discussion

Integration-Kid has been tested by a small number of individual case studies to conduct system verification and revision, but has not been fully tested or evaluated in teaching trials by human students. However, we have discovered that the students seem to be very curious about the companion's response and pay more attention to it than the teacher's. This reinforces the recent research concern and the need of further study of social context and motivation factors in the learning environment of ITS.

5.1 Advantages and Disadvantages of Design Approach

Our approach in designing an LCS is not limited to the domain of integration. For different domains, different curriculum tree structures and protocols of activities need to be constructed. Thus, there will be different rules of behavior or scheduling rules. But the agent production systems, the scheduling facility, the utterance processing, and the screen interface design are all domain independent. For the problem solver, the term rewriting system may be extended to some other domains such as equations of chemical reactions. For other domains, one needs to build different domain problem solvers. Along the implementation, we have constructed an environment for developing curriculum tree, that is, a development shell for LCS. With the use of curriculum tree architecture and situation map, we found that both the implementation time and the complexity of the system are significantly reduced.

However, there are some disadvantages of our design approach. The intended broad coverage of the domain, which starts from concept introduction to complicated miscellaneous problems allows Integration-Kid to simulate a small complete course of integration. While this reflects the implications of peer interaction thoroughly in different stages of learning of the subject, there seems to have not enough attention been paid to peer learning interaction on complex problems. Better focus may be attained if the test domain is limited to miscellaneous problems only.

The use of situation map in Integration-Kid is not comprehensive. For example, the map may cause a learner to give up the current choice and back up for another while it is still possible to carry on with the current problem state though it is inferior. However, we feel that the effort of seeking representations to capture changing problem situations and to embody social texture in a learning environment is still in an early stage. Neither problem solving ability nor natural language understanding techniques in the state of art can simulate problem content particularities and various human individual intuition on problem situations. Indeed, whether learning with social context or situated learning, recently advocated by researchers such as Brown and Collins (Brown et al., 1989; Brown, 1990), is successful, depends upon such representations.

Currently, we have modeled the learning companion as a typical average student with the assumption that the potential users of the system are students of average level. It is desirable that the companion's "level of expertise" can be dynamically adjusted to the performance of the student.

5.2 Limitations of LCS

Several limitations of LCS which are domain independent have been noted. First, there is a possibility that the student does not believe that the computer companion behaves as his classmate. Second, the implementation of LCS is more complex than traditional ITS for the need to model an additional agent, the companion. Third, some of the companion's mistakes might confuse the student. It is necessary to decide what kind of companion's mistakes will or will not confuse the student in order to model the companion. Fourth, from a human-computer interface point of view, it is simpler for the student to interact with one agent rather than two agents. Utterances on multiple panes may cause distraction and difficulty for the student to communicate with the system. Finally, at different stages of learning, different protocols of activities and their implementation in LCS are an extra burden while the two agent model of ITS usually uses the same format of learning activities.

5.3 Perspectives of LCS Research

Despite of the above limitations, we believe that the LCS research will spawn a lot of studies of cognition and learning (Chan & Baskin, 1990). In principle, it is possible to introduce a learning companion to a tutoring system on any domain. Indeed, the paradigm of LCS represents a broad spectrum of ITS design due to the possible varieties on the number and the identities of the agents in an LCS. Each of these varieties gives rise to particular cognitive issues in the student's learning. A project underway is a system of two human students who learn together through a network and the computers are simulated as an intelligent teacher who monitors the learning activities. Such a distributed learning companion system will avoid the first two limitations discussed above.

An extrapolation of LCS research will perhaps be the indication that many computer systems should consider to maintain an explicit companion model rather than an implicit user model in order to enhance the system's human affinity to users.

Acknowledgements

I would like to thank Arthur Baskin, Richard Dennis, Howard Aizenstein, and Lisa Chiu for their support during various stages of this research.

References

Brown, J. S. (1990). Toward a new epistemology for learning. In C. Frasson, & J. Gauthiar (Eds.), *Intelligent tutoring systems at the crossroad of AI and education,* Norwood, NJ: Ablex Publishing Inc.

Brown, J. S., Collins, A., & Duguid, P. (1989). Situated cognition and the culture of learning. *Education Researcher,* Vol. 18, no. 1, pp. 32-42.

Chan, T. W. (1989). Learning Companion System. Ph.D. Thesis, Department of Computer Science, University of Illinois at Urbana-Champaign.

Chan, T. W. (1991a). Curriculum Tree: A Knowledge-Based Architecture for Intelligent Tutoring Systems. (in preparation).

Chan, T.W. (1991b). Some Techniques in Building ITS in Mathematics Domain. (in preparation).

Chan, T. W. & Baskin, A. B. (1988) "Studying with the Prince: The Computer as a Learning Companion." International Conference of Intelligent Tutoring Systems, 1988, June, Montreal, Canada, pp. 194-200.

Chan, T. W. & Baskin, A. B. (1990) "Learning Companion Systems." In C. Frasser & G. Gauthier (Eds.) Intelligent Tutoring Systems: At the Crossroads of Artificial Intelligence and Education, Chapter 1, New Jersey: Ablex Publishing Corporation.

Clancey, W. J. (1986). Qualitative student models. In Traub, J. F. (Ed.), *Annual Reviews of Computer Science,* Vol. 1, pp. 381-450. Palo Alto, CA: Annual Reviews, Inc.

Collins, A., Brown, J. S., & Newman, S. E. (1989). Cognitive apprenticeship: Teaching the craft of reading, writing and mathematics. In L. B. Resnick (Ed.), *Knowing, learning, and instruction: Essays in honor of Robert Glaser,* Hillsdale, NJ: Lawrence Erlbaum Associates Publishers.

Cumming, G., & Self, J. (1989). Collaborative Intelligent Educational Systems. In D. Bierman, J. Breuker, & J. Sandberg (Eds.), *Artificial Intelligence and Education*, pp. 73-80, Amsterdam: IOS.

Dillenbourg, P., & Self, J. (1991). Designing Human-Computer Collaborative Learning. In C. O'Malley (Ed.), Human-Computer Collaborative Learning, Berlin: Springer-Verlag (to appear).

Gilmore, D., & Self, J. (1988). The application of machine learning to intelligent tutoring systems. In J. Self, (Ed.) *Artificial Intelligence and Human Learning, Intelligent computer-assisted instruction*, pp. 179196, New York: Chapman and Hall. Newman, D. (1989). Is a student model necessary? Apprenticeship as a model for ITS. In D. Bierman, J. Breuker, & J. Sandberg (Eds.), *Artificial Intelligence and Education,* pp. 177-184, Amsterdam: IOS.

Piaget, J. (1965). *The moral judgment of the child.* New York: The Free Press.

Self, J. (1988). Bypassing the intractable problem of student modelling. *International Conference of Intelligent Tutoring Systems,* 1988, June, Montreal, Canada, pp. 18-24.

Vygotsky, L. (1978). *Mind in society* . (M. Cole, V. John- Steiner, S. Scribner, & E. Souberman, Trans.) Cambridge, MA: Harvard University Press.