

# Abstract Syntax Tree (AST) Infrastructure in Problem Solving Research

JOSEPH M. SCANDURA\*

*MERGE Research Institute  
Visiting Research Professor, Drexel University and  
Adjunct and Emeritus Professor, University of Pennsylvania*

Both top-down and bottom-up approaches to problem solving face major hurdles in converting theory to working technology. Top-down efforts must do a better job of converting high-level conceptualizations into working systems. Working systems often bear an indirect and non-exclusive relationship to the theories that motivated them. Bottom-up methods need the reverse – insuring that working systems accurately reflect coherent abstractions.

This article shows how Structural Learning Theory (SLT) generally and Abstract Syntax Trees (ASTs) specifically provide a rigorous, systematic way to bridge the gap. SLT puts the focus on higher order knowledge (SLT rules), a universal goal-switching control mechanism and a systematic method for structural (cognitive task) analysis for identifying higher (and lower) order SLT rules. ASTs make it possible to represent declarative (structural) and procedural knowledge simultaneously at ALL levels of expertise (i.e., knowledge).

The article concludes with implications of SLT and ASTs for research highlighted in this issue. A key question is the extent to which pre-analysis of content domains would eliminate ambiguity and increase the precision with which one can predict behavior. Arbitrary irreducible relations, for example, can be refined systematically, and introducing higher order AST-based rules can dramatically reduce the number of nodes required in any given representation.

*Keywords: Structural Learning Theory, Abstract Syntax Trees, Problem Solving, Production Systems, rules, instructional theory.*

---

\*Corresponding author: [scandura@scandura.com](mailto:scandura@scandura.com)

## **BACKGROUND**

Theory development in science typically starts from the top-down, partitioning complex phenomena into “bite-sized” pieces that can more easily be understood (e.g., plant and animal kingdoms in biology and zoology, periodic table in chemistry). Theory development in learning has followed a similar pattern (e.g., Bloom’s taxonomy; Gagne’s conditions of learning).

As a science matures, attention turns to fundamental mechanisms underlying observables (e.g., cell division in biology, mechanisms underlying chemical reactions, role of gravitation in planetary motion). Education over the years has relied similarly on a broad range of psychological theories (e.g., S-R theory, information processing).

As mechanisms become better understood, qualitative mechanisms are superseded by formal theories (e.g., Newtonian physics, quantum theory, string theory). Aside from specialized mathematical models of limited value in education (e.g., stochastic theories of S-R learning, Greeno & Scandura, 1964), the closest instructional sciences have come to formal theories of reasonable scope is the increasingly widespread use of technology.

This article positions the role of technology in problem solving, shows how Abstract Syntax Trees (ASTs) provide a way to bridge the gap between high level conceptualizations and formal mechanisms and outlines the use of ASTs in knowledge representation and Structural Learning Theory (SLT). Finally, I describe implications of ASTs and SLT for research highlighted in this issue.

## **TOP-DOWN VS. BOTTOM-UP APPROACHES TO PROBLEM SOLVING**

Top-down taxonomic and bottom-up infrastructure both coexist in learning problem solving by computer. What does each approach have to contribute, and where are we in the process of transitioning?

Taxonomies and conceptual frameworks have long played a major role in the instructional sciences. They are easy to understand and have proven useful in practical applications. Gagne’s “Categories of learning” (1985), for example, does a good job explicating principles for teaching simpler types of knowledge – his principles have had broad application.

This work has had less impact in complex problem solving. Nonetheless, taxonomic explanation in problem solving has recently attracted attention.

Typologies again may prove useful in identifying the kinds of problems frequently met in education. It is too soon to say, however, how useful they will be in teaching problem solving with computers. Assuming continuity with earlier research based on categories of learning (e.g., XAIDA), problem-solving taxonomists presumably would want to build authoring and/or delivery shells for each kind of problem solving. **All taxonomies, however, inevitably include overlapping categories and/or leave major gaps.** Such limitations become apparent whenever high-level frameworks must be realized as automated (executable) systems. There is the large gap between general guidelines and the detailed decision-making necessary to implement guidelines as executable software. Decisions programmers make during implementation routinely have as much, or more effect on learning than conceptual distinctions guiding the development.

In contrast, fundamental mechanisms are at the forefront in Intelligent Tutoring Systems (ITS research). Efforts to uncover basic learning and instructional mechanisms, however, have had only limited success in ITS (Anderson, 1988). Production systems, for example, promise a deep level of understanding. Their use, however, has had little impact in education. Progress has been slow and arduous over decades, and development remains expensive and time consuming. Why?

The history of science provides clues. A good theory should be precise, operational and preferably general and coherent. Production systems are effectively computer programs and certainly precise. Moreover, like S-R theories, production systems work reasonably well with certain kinds of problem solving (e.g., involving forward or backward chaining). Simply changing the order of productions, however, can dramatically effect behavior. **Theoretical variations have proven difficult to distinguish,** forcing investigators to resort to observation, model tracing (cf. Ritter, 2005) and hard to quantify verbalization techniques.

In short, both top-down and bottom-up approaches to problem solving instruction face major hurdles in converting theory to working technology. **Top-down efforts must do a better job of converting high-level conceptualizations into working systems. Bottom-up methods need the reverse – insuring that working systems accurately reflect coherent abstractions.**

Any viable approach to problem solving instruction must support: a) context-dependent (so-called situated) and context independent higher as well as lower order knowledge, b) individual differences, including differences between novice, neophyte and expert learners, c) relationships between declarative and

procedural knowledge, d) collaborative as well as individual learning and e) ill-defined problem solving. Given the range of problems to be addressed, it is not surprising that investigators are pursuing technical approaches to problem solving from so many different research perspectives (this issue).

### **ABSTRACT SYNTAX TREES (ASTS)**

While routinely used in such well-developed fields as compiler theory and software engineering, ASTs have not been heavily used in Instructional Technology (IT). Nonetheless, ASTs offer major advantages over production systems, directed graphs (originally used in SLT, e.g., Scandura, 1971, 1973, 1977) and relational networks used by other theorists (cf. Scandura, 2005, Table 1). ASTs provide a precise, explicit and general way to close the gap between high-level theory and executable software (Scandura, 2003). Specifically, ASTs make it possible to represent arbitrary concepts and processes in whatever degree of detail might be desired (Scandura, 2003, 2005).

ASTs were built with knowledge representation, learning and instruction explicitly in mind. Given their dynamic hierarchical nature, ASTs and its associated High Level Design (HLD) language (e.g., Scandura, 2003) help theorists bridge the gap from theory to implementation — from conceptualizations that are often little more than vague prescriptions to the realization of those conceptualizations as working systems. The gap at present is so large in most cases that working systems bear at best an indirect and non-exclusive relationship to the theories that motivated them.

### **ROLE OF ABSTRACT SYNTAX TREES (ASTS) IN STRUCTURAL LEARNING THEORY**

Devising a general, coherent and operational theoretical framework for addressing a broad range of relevant phenomena has been central to Structural Learning Theory since the early 1970s (e.g., Scandura 1971, 1973, 1974, 1977). As in contemporaneous research by Newell and Simon in the early 1970s, rule learning and problem solving was a major emphasis in SLT from its inceptions (e.g., Scandura et al 1964, 1969, 1970). Rather than means-end analysis, forward and backward chaining, conflict resolution and other fixed mechanisms in production system models,

however, **SLT put the focus on higher order knowledge (SLT rules), a universal goal-switching control mechanism and a systematic method for structural (cognitive task) analysis for identifying higher (along with lower) order knowledge, including higher order selection rules.** Both ill-defined problem solving and conflict resolution were handled via higher order knowledge (e.g., Scandura, 1971, 1973). These SLT constructs, however, were difficult to realize computationally.

Lacking in those early days was the precision and rigor necessary to convert SLT into executable software — not that we didn't try, and in some cases even come close (e.g., Scandura et al, 1977, 1987). Even more than computing power, the most critical limitation was the lack of needed software tools and infrastructure. Accordingly, while basic research on SLT continued after the late 1970s — albeit at a less intensive and visible level — much of the next 20 years was spent refining the theory and applying SLT principles in the dynamic field of software engineering.

Returning to ITS development in the new millennium (motivated in part by three grandsons and a granddaughter — all under four), I came armed with a fundamentally better way to represent knowledge – SLT rules — in terms of Abstract Syntax Trees with semantic attributes (ASTs), together with a powerful new set of software tools based on ASTs. Each SLT rule includes both data and process, represented respectively by a structure AST and a process AST. Scandura (2003, 2005) further shows that: a) any declarative (structural) or procedural AST may be refined arbitrarily using only three kinds of refinement and b) there is a direct relationship between structures and processes.

Knowledge in SLT was originally represented in terms of higher as well as lower order rules represented as flow charts (directed graphs in which links are operations rather than relations). Directed graphs have important advantages over production systems in representing individual differences. Under laboratory conditions experimental predications approached rarified levels of precision up to 100% (e.g., Scandura, 1971, 1973). Because directed graphs were realized as flow charts, however, rather than executable programs, early ITS research by Brown and his colleagues (e.g., Brown & Burton, 1975) used the “bug” metaphor in programming (including overlays, perturbations and similar terminology) for this purpose with only passing recognition of this earlier work. Similarly, Newell and Simon's work began with an emphasis on programs as performance theory, eventually settling on production systems in the 1970s as the preferred method of representation. Anderson and his colleagues subsequently extended the use of production systems to ITS.

In contrast, ASTs make it possible to represent ALL levels of expertise (i.e., knowledge) simultaneously (see Fig. 1 in Scandura, 2005). ASTs provide:

- a uniform way to represent declarative knowledge (e.g., domain dependence) as well as procedural knowledge — like productions, directed graphs had domains, but there was no simple and heuristically pleasing way to represent data structures in either. Surely data types were not it.
- a simple, uniform and natural way to represent differences between neophyte and expert knowledge. Neophyte knowledge is relatively speaking more procedural, and expert knowledge relatively more structural — with ASTs representing all variations in between. The equivalence between different levels of abstraction in ASTs provides a direct basis for making inferences about what the learner knows (Scandura, 2005).
- a uniform way to represent domain dependent and domain independent knowledge as ASTs. Naïve problem solving requires higher order knowledge, which also is represented as ASTs (which contain other ASTs in their domains and/or ranges). Domains of applicability of such higher order ASTs are themselves ASTs, which easily accommodate all manner of situated variations.

Although ASTs enable more rigor and precision, SLT's major features (Scandura, 1971, 1973, 1977, 2001) remain the same conceptually (at a high level of abstraction) – a systematic method for eliciting knowledge representations, including higher order knowledge, a universal control mechanism, fixed processing constraints for each individual [processing capacity & characteristic processing speed] and a relativistic approach to (individual) knowledge assessment (subsequently referred to as “overlays”).

Key advances beyond the scope of this note are: a) a long sought solution to the fundamental problem of how to **completely separate control from higher order knowledge** and b) a **systematic and largely automated method of knowledge elicitation called Structural Analysis (SA** — realized in AutoBuilder, Scandura, 2005) for creating AST-based SLT rules. Patented methods also are available for extending SA to higher order ASTs, associated with any given problem domain (U.S. Patent, 6,275,976, Scandura, 2001). Higher order AST-based rules and universal control play a central role in Part 2 in a planned series on SLT and AuthorIT (see Scandura, 2001, 2005).

While SLT has only been applied to individuals, the basic theory appears equally applicable to collaborative as well as individual learning

(including ill-defined problem solving – e.g., see futures in Scandura, 2005). Perhaps the single most important benefit of SLT, however, is that it offers a rigorous integrative foundation for various research perspectives, ranging from high level conceptualizations to executable systems.

## **IMPLICATIONS FOR HIGHLIGHT ARTICLES**

What follows are selected implications of SLT and AST-based knowledge representation (SLT rules) for other research highlighted in this issue.

The central point of Scandura's LO article (this issue), for example, is to demonstrate that what a learning object is depends inextricably on the assumed pedagogy (cf. Friesen, this issue) and the way knowledge is represented. Hummel and Koper's (this issue) emphasis on learning objects as processes also is consistent with AST-rules, although the latter necessarily include both procedural and declarative knowledge in varying degrees.

Paquette and Marino (this issue) present a more detailed analysis of learning objects and their relationship to knowledge representation. Particularly relevant to the present analysis are their comments on representing knowledge as trees, and specifically their caution that "tree organization is too restrictive for describing the rich network of relations that ties the concept structures."

The statement is true. However, the fact that any relation is precisely equivalent to an operation changes the situation fundamentally. Specifically, instead of representing the indefinitely varied relationships between elements explicitly, one can instead refine the parent node into an operation. The operation, in turn, can be refined systematically as with any other. Relationships do not lend themselves to systematic analysis (e.g., SA).

The essential point is that ANY idea (node) in a tree can ALWAYS be further refined as detailed in Scandura (2003, 2005) via three basic kinds of structural refinement: component, category and dynamic. Procedural trees may be constructed similarly. Each structural refinement type corresponds to a unique procedural refinement (parallel or loop, selection and interaction/callback). The need for relations is explicitly accommodated via dynamic refinements, thereby avoiding entirely the need for an arbitrary number of relations (which do NOT readily lend themselves to further refinement).

Scandura (2005), for example, shows how dynamic refinements naturally arise even in highly procedural tasks, such as basic arithmetic operations

(e.g., column subtraction). In this case, individual digits (complex relationships among straight and curved lines) are normally considered atomic (i.e., indivisible). Nonetheless, each digit can be represented more precisely as a procedure for constructing it. Each such procedure is a prerequisite that can be refined to whatever level of detail may be appropriate for the intended learner population.

Although typically not distinguished in relational representations, it is important to note that **relationships may exist between trees as a whole as well as within trees**. As a result, the number of relationships goes up geometrically with the number of nodes (cf. Scandura, 2005, p. 211-12). In contrast, relationships between trees in SLT are naturally realized as higher order AST-based rules in which one or more nodes are themselves AST-rules (Scandura, 2001, 2003, 2005). Higher order rules have played a central role in SLT since its inception (e.g., Scandura, 1971). As shown in Scandura (2005, p. 211-12), **modularizing knowledge in terms of higher and lower order AST-based rules dramatically reduces the number of nodes required in any given representation**.

It also is worth noting that **different levels of expertise (knowledge mastery) are represented via different levels of abstraction in an AST representation**. By definition, for example, higher levels of abstraction in an AST effectively ignore lower level detail. Lower level detail only becomes apparent at lower levels in the hierarchy. Higher levels of detail are reflected by the kinds of measures suggested by Paquette and Marino. For example, “knowing” an AST (node) at a high level of abstraction means being able to perform the task more easily and quickly, with greater variety of means, than being able to perform the same behavior in a more prescribed manner (as dictated by lower levels in the hierarchy).

Mitrovic, Ohlsson & Martin (this issue) base their research on Constraint Based Modeling (CBM ) theory in which both declarative and procedural knowledge play a central role. Declarative and procedural knowledge are sharply delineated but treated differently in CBM. On the other hand, AST-based knowledge rules — like all executable programs — necessarily include both declarative (data structure) and procedural aspects, albeit in different degrees. Knowledge structures correspond to observable behavior (cf. data), and procedures to processes for generating that behavior. Although CBM does not teach procedural knowledge directly, CBM imposes constraints (representing declarative knowledge) on the use of procedural knowledge.

These constraints define equivalence classes of problem states, and in

that sense are directly analogous to higher-level nodes in an AST based representation (where lower level details are ignored). Conditions in procedures effectively are constraints that serve to direct flow of control. Other relationships are a logical consequence. Attaching feedback to constraints, for example, corresponds directly to attaching feedback (as well as instruction/hints) to nodes in ASTs. Incidentally, it is worth noting that self-explanation (common in CBM and other ITS systems) involves AST-based rules in which the data structures represent constructs to be explained and their corresponding explanations.

Jonassen (this issue) describes how problem solving can be facilitated by reusing examples (cases). He does a good job of identifying ways that analogical reasoning may be used toward this end -- by inferring generalized solution methods from examples and by generalizing problems themselves. He then describes various techniques that have been used to promote such generalization: questioning, verbal protocols and structure mapping.

This kind of analysis helps identify relevant phenomena, as well as general guidelines for improving problem solving instruction. What it does NOT do is detail underlying infrastructure (i.e., precisely represent what needs to be learned in a form necessary for automation) as is the case with ASTs. Once one has identified what needs to be learned, the question of how to teach it becomes secondary. As Roughead and Scandura showed long ago (1968), what is learned in mathematical discovery can (at least sometimes) be identified, and if so, can be taught even more efficiently via direct verbal instruction. Furthermore, modern Structural Analysis (SA) of given problem solving domains provides a systematic method for identifying what needs to be learned (including higher order knowledge) to solve given classes of problems at whatever level of detail may be desired.

Jonassen's "structure mapping" comes closest to identifying what is to be learned. As above, however, relational networks generally (whether or not they result from structure mapping) have inherent limitations: Among other things, they do not represent arbitrary levels of expertise and do not distinguish different kinds of refinement having direct implications for solution processes (for details, see Scandura, 2003, 2005).

Moreover, prior work (now commonly accepted) conclusively demonstrates that what must be learned necessary depends on the domain in question. As Jonassen notes in his introduction, requisite problem solving skills may involve more than just learning from examples. Scandura (1971), for example, shows that problem solving in mathematics may involve any combination of six basic kinds of mathematical processes: induction from

examples (case based reasoning) & its opposite of constructing examples from generalizations; understanding mathematical descriptions and its opposite of representing mathematical ideas symbolically (and/or as icons); deduction and its opposite of axiomatization.

In a similar vein, Foshay (this issue) analyzes, contrasts and proposes various design dimensions involved in producing instructional simulations. Critical in this respect is the role of representation. While simulations may and properly should include both procedural and declarative knowledge, emphasis on fidelity (with the real world) is often taken to mean modeling directly observable (e.g., visual) aspects of reality. Accordingly, AST data structures play a prominent role. Any real world scenario can be represented in terms of ASTs in whatever level of detail may be desired and/or required.

It is important to understand in this respect the role that different kinds of refinement play in this process (especially see Scandura, 2003). One starts with high-level static abstractions representing the real world concept, or scenario in question. This abstraction is broken down successively into components, categories or operations — as far as necessary to achieve the desired level of fidelity. It is particularly important to notice that dynamic abstractions provide an explicit way to represent observable actions.

Many of the other Highlight Articles in this issue involve the use of adaptive scaffolding of one sort or another in problem solving. Conati and Muldner (this issue) stress the use of examples to good effect. Lajoie, Wiseman and Gauthier (this issue) distinguish cognitive skills used by experts and novices. Wolfe, Arroyo, Beal and Murray (this issue) provide an excellent example of the kind of empirical data that is so badly needed and yet so commonly lacking. Azevedo, Cromley, Winters, Moos and Greene (this issue) demonstrate that providing adaptive scaffolding can be effective in hypermedia (as well as other) environments. Winne, Nesbit and Kumar (this issue) describe an interesting tool that allows learners to interact with multimedia content in a variety of ways. GStudy also tracks and guides learner behavior. Feng and Heffernan (this issue) describe a useful reporting system for teachers. Bill Winn (this issue) describes complexities in gathering and interpreting simulation data. Given contemporary emphasis on group results, it was interesting to find that while even dyslectic and normal children performed similarly as a whole, individual differences were much greater.

A key question in all of these studies is the extent to which pre-analysis of the content domain may help to understand individual differences, eliminate ambiguity and increase the precision with which one can predict behavior. Such analysis may be done informally via operational descriptions

and/or examples, as we did back in the 1960s, more precisely via directed graphs (flow charts), as in the 1970s, or finally and most rigorously in terms of AST-based SLT rules. From an instructional perspective, **perhaps the single most important result of SLT research over a 40 plus year period is that the more precisely one can identify what must (or might) be learned in any given situation, the better job one can do of assessing what the learner does and does not know — and of teaching it.** It is encouraging to know that there are now very explicit methods of structural (cognitive task) analysis for identifying AST-based rules with whatever degree of precision may be desired, as well as tools (AutoBuilder) that automate many of the tasks involved.

Haider and Frensch (this issue) address the central issue of how students learn the higher order skills necessary in problems solving. They found that subjects discovered a general principle when stimulated to do so in one way or another. What is particularly interesting about these studies is that they ask the same question that motivated development of the SLT many years ago. Rather than just indirectly motivating learner discovery, **the above analysis raises the question of what would happen if one were to identify precisely what needs to be learned in given domains in order to make such discoveries in the first place?** Early research on higher order rule learning (e.g., Roughead & Scandura, 1968; Scandura, 1974) provides rather definitive answers. Higher order rules for deriving solution procedures were identified and taught directly in both studies to good effect. The latter study further demonstrated that given suitable experimental controls, it is possible to predict the behavior of individual students on specific problems with deterministic precision (i.e., every subject perform as expected). This result held for far as well as near transfer.

Zheng, Miller, Snelbecker and Cohen (this issue) are concerned with relations between interactive media, spatial abilities and memory load and their influence on problem solving. In this context, it may be noted that memory load is a function not only of the number of elements but the cumulative effects of chunking processes (e.g., Voorhies & Scandura, 1977, Chapter 7). **In AST representations chunking corresponds precisely to transitions between different levels of abstraction.**

In addition to Winn, two highlight articles deal explicitly with individual differences. Shute and Underwood (this issue) identify and summarize limitations of well-known diagnostic approaches to mathematics problem solving, and propose Evidence Centered Design (ECD) as one way to address these limitations. As the author notes the ECD approach shares certain things in common with AST-based representations: Although necessarily subject to

irreducible relationships, the ECD proficiency model also is hierarchical in nature. In addition, both ECD and SLT rules represent local and global proficiency – in the latter case via lower and higher order SLT rules, respectively.

The major difference is that AST-based rules come with an associated formalism, including a small, exhaustive set of structural/declarative and procedural refinement types, plus an explicit method of structural analysis (Scandura, 2001, 2003). The recent introduction of dynamic (structure) refinements and their procedural equivalents (interaction/callbacks) makes it possible for the first time to represent knowledge in a completely arbitrary manner with whatever degree of precision may be desired. Accordingly, proficiency may (but need not) be determined with equivalent levels of precision (e.g., compare early research in mathematical diagnosis by Durnin & Scandura, 1963, diagnosis and instruction in complex problem solving, including both higher and lower order knowledge, by Scandura, 1974, Scandura et al, 1977 and Wulfeck & Scandura, 1977).

Similar comments can be made with respect to Spector, Dennon and Kozalka (this issue). The incremental growth in problem solving abilities over time was the focus of and demonstrated with considerable precision by Wulfeck & Scandura (1977). Because this early research was based on rules represented as flow charts (rather than ASTs), it is interesting to note that Spector et al's subjects reported an initial preference for flow charting

## REFERENCES

- Anderson, J.R. The expert model. In Joseph Psotka, L. Dan Massey and Sharon A. Mutter (Eds.) *Intelligent tutoring systems: lessons learned*. Hillsdale, NJ: Lawrence Erlbaum Associates, 1988, pp. 21-53.
- Brown, J. S. and Burton, R.R Multiple representation of knowledge for tutorial reasoning. In D. Bobrow & A. Collis (Eds.) *Representation and understanding: Studies in Cognitive Science*. New York: Academic Press, 1975, pp. 311-349.
- Gagne, R. *Conditions of Learning* (4th ed.). New York: Holt, Rinehart & Winston, 1985.
- \*\* Ritter, S. Authoring model-tracing tutors. *Technology, Instruction, Cognition & Learning (TICL)*, 2005, 2, 3, 231-248.
- Gagne, R.M. *Conditions of Learning*. Holt, Rinehart and Winston, 1977.
- Roughead, W. G. and Scandura, J.M. "What is learned" in mathematical discovery? *Journal of Educational Psychology*, 1968, 59, 283-298.
- Scandura, J. M. An analysis of exposition and discovery modes of problem solving instruction. *Journal of Experimental Education*, 1964, 33, 149-159. (Reprinted in P. C. Burns & A. M. Johnson (Eds.), *Research in elementary school curriculum and organization*. Rockleigh, N. J.: Allyn and Bacon, 1969.)

- Scandura, J.M. New directions for theory and research on rule learning: II. Empirical Research. *Acta Psychologica*, 1969, 29, 101-133.
- Scandura, J.M. The role of rules in behavior: Toward an operational definition of what (rule) is learned. *Psychological Review*, 1970, 77, 516-533.
- Scandura, J.M. Deterministic theorizing in structural learning: Three levels of empiricism. *Journal of Structural Learning*, 1971, 3, 21-53. (c)
- Scandura, J.M. Mathematics: Concrete behavioral foundations. New York: Harper & Row, 1971.
- Scandura, J.M. *Structural learning I: Theory and research* London/New York: Gordon & Breach Science Publishers, 1973.
- Scandura, J.M. The role of higher-order rules in problem solving. *Journal of Experimental Psychology*, 1974, 120, 984-991.
- Scandura, J.M., Durnin, J.H., & Wulfeck, W.H., II. Higher-order rule characterization of heuristics for compass and straight-edge constructions in geometry. *Artificial Intelligence*, 1974, 5, 149-183.
- Scandura, J.M., Wulfeck, W. H., II, Durnin, J. H., & Ehrenpreis, W. Diagnosis and instruction of higher-order rules for solving geometry construction problems. In Scandura, J.M. *Problem Solving: a Structural / Process Approach with Instructional Implications*. New York: Academic Press, 1977.
- Scandura, J.M. & Scandura, A.B. *Structural Learning and Concrete Operations: An Approach to Piagetian Conservation*. Praeger, N.Y., 1980.
- Scandura, J.M. Problem solving in schools and beyond: Transitions from the naive to the neophyte to the master. *Educational Psychologist*, 1981, 16, 139-150.
- Scandura, J.M. A structured approach to intelligent tutoring. Chapter 14 in D.H. Jonassen (Ed.) *Instructional design for microcomputer software*. Hillsdale, N.J.: Erlbaum, 1987, 347-379.
- \* Scandura, J.M. Structural learning theory in the year 2000. *Special Monograph of Journal of Structural Learning and Intelligent Systems* (a special monograph), 2001, 14, 4, 271-305. (Similar version sans Appendices in Scandura, J.M. *Structural Learning Theory: Current Status and Recent Developments*. *Instructional Science*, 2001, 29, 4, 311-336.)
- \*\* Scandura, J.M. Domain Specific Structural Analysis for Intelligent Tutoring Systems: Automatable Representation of Declarative, Procedural and Model-Based Knowledge With Relationships to Software Engineering. *Technology, Instruction, Cognition & Learning (TICL)*, 2003, 1, 1, 7-57.
- \*\* Scandura, J.M. AuthorIT: Breakthrough in authoring adaptive and configurable tutoring systems.. *Technology, Instruction, Cognition & Learning (TICL)*, 2005, 2, 3, 185-230.
- Voorhies, D. & Scandura, J.M. Determination of memory load in information processing. In Scandura, J.M. *Problem Solving: a Structural / Process Approach with Instructional Implications*. New York: Academic Press, 1977.
- W.H. Wulfeck, & J.M. Scandura. Xxxx Chapter 14 in Scandura, J.M. *Problem Solving: a Structural / Process Approach with Instructional Implications*. New York: Academic Press, 1977.
- \* On-Line at: [www.scandura.com](http://www.scandura.com) under TICL On Line
- \*\* On-Line at: <http://www.oldcitypublishing.com/TICL/TICL%201.1%20contents.html>
- Friesen, N. *Technology, Instruction, Cognition & Learning (TICL)*, 2005, 3, 1, this issue.