Learning-Based Constraints on Schemata[1]

Peter C.R. Lane (pcl@psychology.nottingham.ac.uk)
Fernand Gobet (frg@psychology.nottingham.ac.uk)
Peter C-H. Cheng (pcc@psychology.nottingham.ac.uk)
ESRC Centre for Research in Development, Instruction and Training,
School of Psychology, University of Nottingham,
University Park, NOTTINGHAM NG7 2RD, UK

Abstract

Schemata are frequently used in cognitive science as a descriptive framework for explaining the units of knowledge. However, the specific properties which comprise a schema are not consistent across authors. In this paper we attempt to ground the concept of a schema based on constraints arising from issues of learning. To do this, we consider the different forms of schemata used in computational models of learning. We propose a framework for comparing forms of schemata which is based on the underlying representation used by each model, and the mechanisms used for learning and retrieving information from its memory. Based on these three characteristics, we compare examples from three classes of model, identified by their underlying representations, specifically: neural network, production-rule and symbolic network models.

Introduction

One of the unifying themes in cognitive science is the use of *schemata* for explaining the units of knowledge within humans. However, the specific properties which comprise a schema usually vary between authors. Early work in the AI and cognitive traditions (e.g. Rumelhart, 1980) set the scene for the use of schemata in computational models of learning. It is now appropriate, with a number of successful models in the literature, to see what forms of schemata arise within a learning-based system. This question is especially interesting because computational models do not simply implement basic concepts such as schemata with an added learning mechanism. Instead, each computational model is based on some core representational structure and primitive learning mechanisms, from which structures such as schemata may be inferred.

The aim of this paper is to consider examples from a number of computational models and simply extract those elements which most relate to schemata. The difficulty here is that the models have not been tested on identical tasks, and so the comparison must be at a more qualitative level. Hence, we begin with some informal definitions of schemata to define our analytical framework.

Learning and Using Schemata

Brewer (1999) defines schemata as "the psychological constructs that are postulated to account for the molar forms of human generic knowledge." The idea is that knowledge of visual scenes or discourse structure may be considered in terms of basic units. For instance, house-scenes typically consist of rooms, each room containing certain basic properties, such as walls or furniture. The schema for a room will contain *slots* for the properties, and, in the absence of specific information, these slots will be filled with *default* values. So a room will, by default, be considered to have four walls, a ceiling, a door, lighting, probably a window, and so forth.

Less committal is the definition by Rumelhart (1980; italics in original): "A schema theory is basically a theory about knowledge. It is a theory about how knowledge is represented and about how that representation facilitates the *use* of the knowledge in particular ways." Rumelhart therefore focuses on the form of the schema theory (representation and reuse), whereas Brewer (1999) defines the form of the schema (a molar form of knowledge). Rumelhart's definition is also echoed in that of Sweller (1988), whose concern is with modelling problem-solving behaviour. According to Sweller (1988), a schema is simply a "structure which allows problem solvers to recognize a problem state as belonging to a particular category of problem states that normally require particular moves. ... certain problem states can be grouped, at least in part, by their similarity and the similarity of the moves that can be made from those states." Each of these definitions stresses the functionality of the knowledge in the schema. Also worth noting is that the schema is a form of retrieval structure, identifying elements from earlier experience which can be reused in the current situation.

Our interest in this paper is in describing computational models of learning, and for this purpose, as will become evident later, a fairly loose definition of schemata is required to provide the basis of comparison between different models. Hence, we will use the following definition:

A schema is a cognitive structure for representing and retrieving classes of typical situations for which a similar response is required of the learner.

Our comparison looks at the variation in schema-form based on the different assumptions underlying each model. The greatest assumption made is the basic *representation* used by the model for storing learnt information in its

[1] In *Proceedings of the Twenty-Second Annual Conference of the Cognitive Science Society*, (pp. 776-81, 2000). Mahwah, NJ: Erlbaum.

memory. This representation may be highly structured, localised or distributed. The type of the representation affects the processes which the model can use to *learn*, where learning is the process of converting what has been experienced into an internal representation. In this context, some representations provide better support for incremental real-time learning, whereas others are better for complex rule induction. The type of representation also affects the *retrieval* of information from the model's memory for use in novel situations. Some systems assume that every item of memory is compared to determine the closest match to the current situation, whereas others maintain a hierarchy for indexing their memory and consequently only search a subset of the total memory.

These three characteristics, for representing, learning and retrieving a schema, provide a framework for analysing how different computational models address the questions of learning and using schemata. We use this framework in the next three sections, where we compare examples from three classes of model. The classes are distinguished by their underlying representations: neural network, production-rule and symbolic network models. The examples are selected to be representative (without attempting to be comprehensive).

Neural network models

The ability of a PDP (Parallel Distributed Processing) model (otherwise known as a neural network) to learn schemata was addressed at an early stage by Rumelhart, Smolensky, McClelland and Hinton (1986), who described how such properties can arise within a class of PDP models. However, they did not address the question of learning. A better demonstration of these ideas within the context of a learning system is the Sentence Gestalt (SG) model of St. John and McClelland (1990). We also consider the CLARION system of Sun, Merrill and Peterson (in press), which is a hybrid model of skill learning.

Sentence comprehension

The aim of the SG model (St. John & McClelland, 1990) is to capture the process by which people fill out semantic information whilst reading a sentence. For example, given the sentence 'Bobby pounded the board together with nails', the inference "with a hammer" is made automatically. We can explain such behaviour by hypothesising that people recall (subconsciously) some schema for the sentence from which default information (the hammer) can be inferred. The SG model attempts to account for such phenomena. It consists of a two-stage recurrent neural network. The first stage learns a distributed representation for the sentence, called the sentence gestalt, from a temporal sequence of constituents. Each constituent is either a simple noun phrase, a prepositional phrase or a verb. The second stage acts as a probe for information contained in the sentence gestalt. Each probe is a role/filler pair, and the sentence gestalt is probed by presenting either a role or a filler, from which the network is to supply the complete pair. Requested information need not refer directly to words in the sentence. For example, after seeing 'Mary ate the spaghetti', the model should return the filler "fork" for the role "instrument".

The experiments performed by St. John and McClelland demonstrate that the SG model successfully assigns constituents to thematic roles based on syntactic and semantic constraints. Further, the model can disambiguate meanings and instantiate vague terms as appropriate to their context and the training data previously seen by the model. This behaviour fulfills the requirements for schemata as discussed previously: the model classifies sentences into various groups, and these groups can have variable or default information associated with them.

We can now consider the schemata used in SG against the three basic characteristics of our framework:

Representing a schema

All knowledge contained within a neural network is held implicitly across the weights within the network. Once activation is presented on the input, every weight and node within the network interact to generate an output. In this situation, specific schemata are not really *represented* within the network, in the sense of identifiable units, but instead *emerge* as a consequence of the specific set of inputs. Hence, the schemata used by the model cannot be extracted for use as explicit rules, but instead must be inferred from their effects on the network's output.

Learning a schema

Given the nature of distributed representations, it is not possible to learn about just one schema, because of the unpredictable effect on other information held in the weights. Indeed, the process by which the SG model (and most similar neural network models) is trained involves continuous passes of the entire training dataset whilst the weights in the network are gradually altered to approximate the mapping between the input data and its target output. This process means that the network captures generalisations true of the entire dataset, making it robust in novel situations.

Retrieving a schema

Again, the nature of the distributed representation within the model implies that the whole network is activated when obtaining a response to a novel input. Hence every piece of acquired information (every weight value) is used in generating a response. This process additionally ensures a robust response in novel but similar situations, because the retrieval process is based on the *similarity* between the novel input and the model's previous experience. For instance, if a large number of examples are presented to the network, and the responses analysed, it will be seen that those examples which are most similar tend to generate similar responses. Conversely, if a novel input is partly similar to one type of example in the training data, and partly similar to another type, the computed response will fall somewhere between that for the two items of training data. Note that the similarity in input to the network is heavily dependent on the form of encoding used for representing each item of data to the network on numeric input units.

Bottom-up skill learning

CLARION (Sun, Merrill & Peterson, in press) is a hybrid model for bottom-up skill learning. It is designed to model the process by which low-level perceptual-motor skills are converted into explicit rules, and also capture the interaction between these two levels of knowledge whilst carrying out a complex task. CLARION assumes that declarative knowledge is represented explicity within a rule-based system, whereas procedural knowledge is represented implicity within a neural network. CLARION has been tested in a perceptual-motor task involving navigation through a minefield, in which the model must learn to react to particular visual patterns of mines with appropriate navigation instructions to avoid the mines and reach a target. The dual use of knowledge is reflected in subjects' responses: mostly they react instinctively, but after some experience in the domain some explicit planning is reported. CLARION's use of two knowledge levels is intended to capture this shift towards more explicit knowledge.

The novelty in CLARION is that the rules can either be pre-programmed (i.e. taught in the standard top-down manner) or learnt based on the low-level knowledge in the neural network. Specifically, if the neural network suggests an action which satisfies its criterion for success, then the current sensory state is turned into the condition part of a new production in the rule set, with its action part being the currently suggested action. Further learning processes on the rules update statistics and may refine and alter rules for efficiency. CLARION therefore contains two independent learning mechanisms, but the two can also work together with an interesting transfer of bottom-up (procedural) knowledge into the explicit rule-set. As with SG, schemata are evident in the similarity-based generalisations made by the model.

Representing a schema

CLARION uses a two-level representational structure: a rule-based system and a feed-forward neural network. As with SG, schemata are seen to emerge through the interaction of many elements in the model. Hence, the network and the rules can generalise robustly to novel situations based on partial similarity. The purpose of the rule-based system is to 'fix' the generalisations learnt by the neural network and prevent later experience 'blurring' them. These rules may in themselves represent broader classes of situation, because some of the attributes can have variables instead of specific values, rather akin to slots on a more generic template.

Learning a schema

The procedural knowledge in CLARION is learnt in a similar manner to the SG model described above, using a modified form of backpropagation: an additional reinforcement term is included in the training error because the correctness of a specific action is only known at the completion of the task. The rule-based declarative knowledge includes mechanisms for constructing new rules, or expanding or shrinking the conditions of existing rules. The mechanism for constructing a new rule is merely to include, for a successful action, the situation and action as the condition and action parts of a new rule. Expanding or shrinking a rule's conditions amounts to increasing or decreasing the likelihood of the rule matching future inputs by altering the range of possible values in one of its attributes. Before making any such changes to a rule's conditions, an information gain for each rule is computed to determine whether a modified version would do better than the current rule.

Retrieving a schema

As with SG, the whole of CLARION's memory is probed simultaneously to determine all information relating to the current situation. The possible actions suggested by the separate procedural and declarative levels are then chosen through a weighted competition, reflecting the degree of emphasis CLARION is placing on each type of knowledge. Note that in both levels CLARION relies on a similarity-based metric to generalise to novel situations. This is natural in the chosen domain, where all inputs are visual scenes; the rules basically contain a localist representation of information similar to that in the neural network.

Summary

The form of schemata possible in these neural network models is determined partly by their learning mechanisms and partly by their retrieval mechanisms. The basic neural network is capable of learning complex mappings from the input to output data, and inherent mechanisms within the neural network are used to retrieve information most similar to the current situation. In CLARION, situations may be learnt explicitly with specific rules consisting of core and variable information.

Production-rule models

Production rules have been a popular representation for a number of computational models, two notable examples being Soar (Laird, Newell & Rosenbloom, 1987) and ACT-R (Anderson & Lebiere, 1998). However, such models are also difficult to discuss in our framework, as their inherent power makes them suitable for application in a wide range of domains and settings, as well as for testing various theories of learning: there are few architectural constraints which have a significant bearing on the forms of knowledge learnt. Here, we describe the generic learning and retrieval mechanisms in Soar.

Soar: chunking of productions

The Soar system integrates perceptual-motor behaviour with basic capabilities for learning and problem solving. All knowledge within Soar is held in the form of productions, with a working memory holding specific attributes and their values. Soar operates in a cycle, attempting to satisfy some goal within its working memory. This cycle takes the contents of working memory and matches it to productions in its knowledge-base. These matching productions place new goals or other elements into working memory (this is known as the *elaboration* phase, which proceeds until all eligible productions have fired, *quiescence*), and then a decision is made as to which of the new goals to pursue next.

Representing a schema

All behaviour within Soar is goal oriented, in the sense that the system is always trying to satisfy some goal or another. Each goal contains three slots: the current problem space, state and operator. The specific representations for information in these slots can vary across applications. A particular schema may not be represented specifically in a production, but instead, in a specific context, a number of similar rules will be matched, suggesting interrelated subgoals, and so yield the effect of a schema.

Learning a schema

Learning within Soar is based on a chunking process that creates new rules. Each rule recreates the results of subgoals in relevantly similar future situations (Laird, Rosenbloom & Newell, 1986). Chunking relies on an analysis of the dependencies within the solution to a given subgoal to create new rules. A new rule is created for each independent result, with a condition relating to the dependency analysis of the subgoal, and an action relating to the specified subgoal. This chunking mechanism is a universal learning mechanism, similar to explanation-based learning (see Rosenbloom & Laird, 1986). The interesting facet of learning within Soar is its ability to focus on those aspects of the situation used for problem solving, and to use only these relevant aspects in chunking. This focus ensures that the chunks learnt by Soar will generalise to novel situations. In addition, Soar has a process of variabilisation, in which information is made as general as possible before it is stored as a chunk in a production.

Retrieving a schema

The retrieval mechanisms within Soar operate only in its elaboration phase, in which “all directly available knowledge relevant to the current situation is brought to bear” (Laird, Rosenbloom & Newell, 1987). In this phase, every production in its memory whose condition directly matches something in the working memory is activated, and its suggested subgoals and other information are added to memory. Matching productions against working memory is based on the similarity of the attributes and their values.

Summary

Just as with neural networks, no specific structure corresponding to a schema exists in Soar. However, the basic learning mechanism within Soar, chunking, does limit the form and content of learnt productions. Firstly, productions are retrieved based on their similarity to items in working memory. The features placed within a production are taken from the set of dependent relations in the attainment of a goal. In addition, some variabilisation can occur on the features.

Symbolic network models

This section considers a pair of models which construct symbolic networks of symbol-level information within a hierarchy. Each of these lays some claims to universality of application, but have currently only demonstrated good results in one or two areas. The first is the CHREST model, which learns about chess patterns, and the second is EUREKA, which learns about physics problems.

CHREST: storing chunks into templates

The CHREST (Chunk Hierarchy and REtrieval STructure) model of expertise (Gobet & Simon, in press) is a recent development of EPAM (Elementary Perceiver and Memoriser) (Feigenbaum & Simon, 1984). The learning processes in EPAM include mechanisms for constructing a discrimination network and incorporating information into it; the learnt information is known as *chunks*. CHREST includes extra mechanisms for learning *templates* (Gobet & Simon, in press); it is this template which is of interest to us here, as it possesses schema-like properties.

A template is created in the following manner. During training, CHREST (just like EPAM) builds a discrimination network of chunks of information. Specific to CHREST is the ability to create lateral links (Gobet, 1996): in this case, *similarity links*. These similarity links can be used whilst searching the network to suggest chunks not directly linked by the tests in the network. However, the novel aspect of this is that a node can reorganise information in similar chunks (satisfying an overlap criterion) into a template. This template contains a core pattern, based on the original chunk, and a set of slots, for the information which varied across the associated chunks.

Representing a schema

CHREST represents all information as chunks within nodes in a discrimination network: a chunk is a familiarised pattern. Nodes are linked by test links, which require some features to be matched on traversal. Some of the nodes in the network contain *templates*, where a template contains a core chunk and a number of slots. However, CHUMP (Gobet & Jansen, 1994) and CHREST+ (Lane, Cheng & Gobet, 2000) additionally allow nodes in the network to be associated with information about possible moves or problem solutions, allowing CHREST to learn to solve problems.

Learning a schema

The discrimination network within CHREST is learnt through four learning mechanisms. Beginning from the root node, CHREST sorts a novel pattern through the network until no further test links can be applied. At the node reached, two things can occur. First, the pattern may match the chunk, in which case more information can be added to the chunk from the pattern (familiarisation). Second, the pattern may mismatch the chunk, in which case a further test link and node are created based on the mismatching features (discrimination). The third learning mechanism constructs *similarity links* between two nodes when their chunks have at least 3 identical items. Finally, for a node with at least 5 similarity links satisfying an overlap criterion, the chunk may be replaced by a template. This template uses the existing chunk as its core, and the varying information across the other nodes as its slots.

Retrieving a schema

Retrieving knowledge within CHREST is achieved simply by following the test links from the root node, applying the tests to the target pattern until no further test applies. The chunk at the node reached is the retrieved schema.

EUREKA: restructuring knowledge

EUREKA (Elio & Sharf, 1990) demonstrates how an effective organisation for large amounts of domain-specific knowledge can support efficient recognition and application of relevant knowledge to the problem at hand. Secondly, the model demonstrates how the qualitative shift from novice to expert levels of knowledge and organisation can arise within a learning framework. EUREKA uses a discrimination network, rather like the CHREST model described above, but instead of simple chunks, the nodes in EUREKA's network hold Memory Organization Packets (MOPs) (Schank, 1980). Each MOP represents a complex knowledge structure holding generalised knowledge extracted from a group of individual experiences. Differences between experiences are encoded in the tests between the links in the discrimination network, and so similar previous experiences are retrieved based on the features in the network which match the current experience.

EUREKA has been applied to physics problems, and is initialised with a set of MOPs containing basic knowledge about physics concepts, equations and inference rules. However, this knowledge does not contain any information about their usefulness or relevance in any particular type of problem. When EUREKA is given its first physics problems, it must use its basic knowledge in conjunction with a means-ends problem-solving strategy to construct a solution. Having done this, EUREKA then places the entire problem and its solution (features, inferences and solution steps) into a P-MOP (Problem MOP). This P-MOP is then stored in the P-MOP network, where some reorganisation of the network may occur. When solving later problems, EUREKA can use information in a P-MOP in preference to its means-ends analysis, which can lead to a shift in EUREKA's problem-solving strategy towards a greater use of important abstract physics concepts, such as force or energy, usually not present in the problem statement. Also, the use of a P-MOP instead of means-ends analysis means the model begins to solve problems working forwards from the given information instead of backwards from the target, in accordance with observed differences between novice and expert problem solvers (cf. Koedinger & Anderson, 1990; Larkin, McDermott, Simon & Simon, 1980).

Representing a schema

EUREKA stores information in a network of P-MOPs. At the root of the network is a P-MOP representing a "generic physics problem". Each P-MOP contains several elements: firstly, a set of *norms* represent the features which a problem must satisfy for this P-MOP to apply; secondly, a set of *indices* (links) to other P-MOPs, with the index specifying the feature(s) which distinguish between them; thirdly, the P-MOP includes a general inference rule; fourthly, the P-MOP includes a specific solution method for carrying out the inference rule; and fifthly, the P-MOP includes a count for the number of problem-solving experiences which it organises (i.e. has matched in the past). The P-MOP representation is a clear example of an explicit schema, with the norms indicating the class of similar problems to which its inference rules will apply.

Learning a schema

EUREKA's learning mechanisms operate through a process of *reorganisation.* Once a problem has been solved, everything about the problem and its solution is collected into a problem-solving experience. This experience is then compared with the existing P-MOP retrieved from the network. If any of the norms differ between the P-MOP and the experience, these are removed from the P-MOP and used as indices to new organisation beneath this P-MOP; any inference rules referring to these differing norms are also removed from the P-MOP and included in the new organisation. This process has the side-effect that partial solution methods may reside on P-MOPs. A further reorganisation can occur in cases where a descendant P-MOP covers most of the problem-solving experiences of its parent P-MOP; in such situations the organisation of the network is not efficient, and one of the discriminating features might be better seen as a commonality.

These two learning mechanisms can lead the network to focus on abstract features in the following way. A property such as a force may not be represented within the problem statement, however, it will be referred to in the problem solution. As problem-solving experiences are gathered, a number will be seen to include force within their solution, and so this feature will become a norm within the P-MOP. From there, the feature may be used to discriminate between different P-MOPs, because it has been derived as a feature of a number of problem-solving experiences.

Retrieving a schema

Each P-MOP in EUREKA's memory is a separate schema, and each is indexed through the P-MOP network. Any of the features in the initial problem representation can serve as indices into the P-MOP network. Whenever the feature appears as a difference in the P-MOP, the corresponding index is traversed. If a number of indices may be traversed, then EUREKA prefers the index leading to the P-MOP that organises the most problem-solving experiences. Hence, EUREKA is directed preferentially to patterns that recur most often. During the traversal, EUREKA will apply the inference rules of any P-MOPs that match the current situation; this process will alter the current situation (the set of equations and unknowns) and so affect the further traversal of the P-MOP network. Note that EUREKA's bias towards P-MOPs which organise larger numbers of problem-solving experiences means that P-MOPs arising from reorganisation of the network will be preferred during problem solving. It is this bias which ensures EUREKA will preferentially use P-MOPs emphasising the presence of forces or abstract entities: as discussed above, such P-MOPs are formed from the aggregate of several more concrete P-MOPs, and so organise a larger number of problem-solving experiences.

Summary

The symbolic network models are closer to the spirit of traditional schemata theories. In particular, there is a close correspondence between the information in a P-MOP or the pairing of problem and solution nodes within CHREST, and the schemata discussed in Koedinger and Anderson (1990). Both models can use information to partially match a current situation. However, different learning mechanisms encode different kinds of information in their nodes; CHREST restricting itself to perceptual similarity, with EUREKA inferring more abstract quantities for use in discrimination.

Conclusion

This paper has taken an inductive approach to the question of how to learn schemata by applying an analytical framework to a number of computational models, and describing the ways in which these models represent, learn and retrieve schemata. Our aim has been to uncover, from existing models, the origins of constraints on the possible forms of schemata. From our analysis we can see some similarities across all the models. Firstly, all use a distributed form of representation, in the sense that schemata for novel situations will usually arise from a number of partial matches, although the symbolic network models possess more explicit schema-like structures. Secondly, all use a similarity-based form of retrieval, differing in the features which may be used for discrimination. In particular, EUREKA allows abstract features (not perceptually obvious) to become significant.

However, the differences in behaviour of the various models are largely down to their specific learning mechanisms. As stated in the introduction, the motivation for these models has not been to learn schemata, as such, but instead to learn effectively in general situations. We therefore conclude that, for the purposes of developing a more meaningful definition of schemata, we should begin by analysing the available range of learning mechanisms in models such as those referred to here. These learning mechanisms should be explored in their cognitive implications. For instance, the use of seriality or resource bounds, the malleability of learnt features and how wide-ranging any changes to previous knowledge may be. Most of these properties will come directly from the learning mechanisms, whereas others will be imposed by the interaction of the learning mechanisms with the other properties of the system, such as its use of perceptual-motor stimuli. Once these properties have been understood, the use of schemata for describing the units of knowledge within humans will become grounded in the processes by which that knowledge has been learnt.

References

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought* (Lawrence Erlbaum).

Brewer, W. F. (1999). Schemata. In R. A. Wilson & F. C. Keil (Eds.) *MIT Encyclopedia of the Cognitive Sciences*, pp. 729-730.

Elio, R. & Scharf, P. B. (1990). Modeling novice-to-expert shifts in problem-solving strategy and knowledge organization. *Cognitive Science*, 14, 579-639.

Feigenbaum, E. A., & Simon, H. A. (1984). EPAM-like models of recognition and learning. *Cognitive Science*, 8, 305-336.

Gobet, F. (1996). Discrimination nets, production systems and semantic networks: Elements of a unified framework. *Proceedings of the Second International Conference of the Learning Sciences* (pp. 398-403). Evanston, III: Northwestern University.

Gobet, F. & Jansen, P. (1994). Towards a chess program based on a model of human memory. In H. J. van den Herik, I. S. Herschberg, & J. W. Uiterwijk (Eds.), *Advances in Computer Chess 7*. Maastricht: University of Limburg Press.

Gobet, F. & Simon, H. A. (in press). Five seconds or sixty? Presentation time in expert memory. *Cognitive Science*.

Koedinger, K. R., & Anderson, J. R. (1990). Abstract planning and perceptual chunks: Elements of expertise in geometry. *Cognitive Science*, 14, 511-550.

Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence*, 33, 1-64.

Laird, J. E., Rosenbloom, P. S., & Newell, A. (1986). Chunking in Soar: The anatomy of a general learning mechanism. *Machine Learning,* 1, 11-46.

Lane, P. C. R., Cheng, P. C-H., & Gobet, F. (2000). CHREST+: Investigating how humans learn to solve problems using diagrams. *AISB Quarterly*, 103, 24-30.

Larkin, J. H., McDermott, J., Simon, D. P., & Simon, H. A. (1980). Models of competence in solving physics problems. *Cognitive Science*, 4, 317-345.

Rosenbloom, P. S., & Laird, J. E. (1986). Mapping explanation-based generalization onto Soar. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 561-567). Philadelphia, PA: MIT Press.

Rumelhart, D. E. (1980). Schemata: The building blocks of cognition. In R.J. Spiro, B.C. Bruce and W.F. Brewer (Eds.) *Theoretical Issues in Reading Comprehension* (Lawrence, Erlbaum), pp. 33-58.

Rumelhart, D. E., Smolensky, P., McClelland, J. L. & Hinton, G. E. (1984). Schemata and sequential thought processes in PDP models. In D. E. Rumelhart & J. L. McClelland (Eds.) *Parallel Distributed Processing, Vol, II.* MIT Press, Cambridge, MA.

St. John, M. F., & McClelland, J. L. (1990). Learning and applying contextual constraints in sentence comprehension. *Artificial Intelligence*, 46, 217-257.

Schank, R. C. (1980). Language and memory. *Cognitive Science*, 4, 243-284.

Sun, R., Merrill, E. & Peterson, T. (in press). From implicit skills to explicit knowledge: A bottom-up model of skill learning. *Cognitive Science*.

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning, *Cognitive Science*, 12, 257-285.