Companion Cognitive Systems: Design Goals and Some Lessons Learned

Ken Forbus, Matt Klenk, Tom Hinrichs

Qualitative Reasoning Group
Northwestern University
2133 Sheridan Rd, Evanston IL 60208
{forbus, m-klenk, t-hinrichs}@northwestern.edu

Abstract

Companion Cognitive Systems is a cognitive architecture inspired by natural intelligent systems. In this paper, we describe seven design goals of Companions, relate them to properties of human reasoning, and discuss their implications. We present our experiences in developing and experimenting with Companions so far, and the challenges that remain.

Introduction

Naturally intelligent systems are organisms. This observation is obvious, but surprisingly, it does not seem to play a central role in most current cognitive architectures. Perhaps even more importantly, all natural intelligent systems we know of are very social organisms. Indeed, many speculate that social behavior, not manipulation or tool-making, has been the major driving force in the evolution of intelligence [Tomasello, 1999]. The Companion cognitive architecture has, as a fundamental goal, understanding how to build intelligent systems that are social beings. Vygotsky [1962] argued that much of our knowledge is learned from interactions with others in our culture, with apprenticeship being especially important. We believe the same approach can be used to create cognitive systems that operate closer to human-level cognition. Companions can be considered as the first attempt to create a Vygotskian cognitive architecture.

This is clearly a very different point in the space of possible cognitive architectures, compared to skill-oriented architectures such as ACT-R [Anderson & Lebiere, 1998], SOAR [Laird, 2008], PolyScheme [Cassimatis, 2006], Icarus [Langley & Choi, 2005], and others. Another significant difference is that we have been motivated by the growing body of evidence that analogical processing is a core operation of human cognition [Gentner, 2003]. Gentner and her colleagues have been amassing evidence that structure-mapping operations occur everywhere from medium-level vision up through conceptual change [Forbus, 2001]. Companions take analogical processing, as defined by Gentner's [1983] structure-mapping theory, as fundamental to its operation.

This paper summarizes the current state of our work on Companions in two ways. First, we describe seven key features of our current design: (1) analogical processing, (2) extensive conceptual knowledge, (3) flexible reasoning, (4) coarse-grained distributed implementation, (5) ubiquitous learning at multiple levels, (6) long-lived continuous operation, and (7) natural interaction. Every implemented cognitive architecture has design decisions that are based on theoretical bets and empirical evidence, and others that are based on engineering concerns. We tease apart these concerns for each of these major choices, and indicate where we have a solid foundation and where things are very much in flux. Second, we summarize a number of recent experiments with Companions and some of the lessons learned from them. We conclude by discussing some of the challenges we face.

Central Design Goals

The Centrality of Analogy

A central hypothesis of Companions is that analogical processing is central to human reasoning and learning [Gentner, 2003]. This is quite a different choice than most cognitive architectures, where analogy is at best relegated to a side role. We discuss the three major processes in turn, focusing on how each is used in the architecture.

Analogical matching is the first operation. The traditional hallmark of analogical matching are distant, cross-domain mappings, like understanding that heat is like water. While cross-domain mappings are indeed important, they are far from the whole story. For example, when you learn that you can start a car by using a key, for instance, you tend to assume that you can start another car by also using a key. These mundane pieces of everyday reasoning appear to use the same mechanism – a within-domain analogy, in this case, from one car to another. Structure-mapping theory postulates that analogy and similarity are based upon structural alignment between representations. This structural alignment is also used for comparison, to understand similarities and differences between two things.

Our model of analogical matching is the Structure-Mapping Engine (SME) [Falkenhainer *et al.*, 1989]. It takes as input two structured representations (*base* and *target*) plus a (possibly empty) set of constraints on the match. It produces one or more *mappings*, each of which has three parts. The *correspondences* indicate what items in the base go with what items in the target. The *structural evaluation score* provides an estimate of match quality. The *candidate inferences* are conjectures about the target, constructed by projecting facts from the base that are only partially mapped. Exploiting analogies is central to Companions reasoning. This is a powerful mechanism for two reasons. First, it can exploit example-specific explanations in new reasoning. Such examples abound in natural communication (e.g., examples in textbooks and fables). Analogical reasoning in the "inner loop" means that one can apply these particular lessons to a range of new situations, without attempting to induce a general rule immediately. Second, it provides an alternative to fine-grained chaining, a process which can easily explode. Importing a whole relational structure is like striding through an inference space with seven-league boots.

Similarity-based retrieval, which finds potentially useful prior experiences, is the second analogical process. Psychological evidence has consistently shown that human similarity-based retrieval is sensitive to surface information, not just deep relational structure. This makes sense if one considers that most analogies made in an organism's daily life are within-domain comparisons. Cross-domain retrievals will be relatively rare, unless there is heavy relational encoding, thereby making more overlap with situations that, on the surface, seem quite different. Again, this is consistent with findings that domain experts tend to have more relational retrievals [Ross, 1989].

Our model of similarity-based retrieval is MAC/FAC [Forbus *et al.*, 1994]. MAC/FAC takes as input a *probe* and a *case library*. It returns one or more cases from the library that (approximately) best match the probe. In Companions, the probe is typically the contents of working memory. A comparison with standard CBR retrieval systems provides a useful contrast. The majority of CBR systems today use feature vector representations, which means that they cannot represent plans, explanations, arguments, or other important aspects of human conceptual structure. Some relational CBR systems still exist, but these rely on hand-coded indexing schemes, which are carefully crafted for specific domains and tasks. MAC/FAC uses relational representations, but does not require any hand-indexing. In Companions, the combination of analogical retrieval and matching has provided a simple but very powerful learning mechanism: Learning by accumulating examples.

The third analogical operation is *generalization*. People are conservative learners, in that they rarely construct an accurate general model with one example. Unlike explanation-based learning [Ellman, 1989], people often don't have complete and correct theories of the domains they deal with, so such caution is wise. On the other hand, they demonstrably learn much faster than today's statistical learning systems do [Wahlster, 2000]. We believe that analogical generalization happens in two circumstances. First, when very similar situations are compared, it appears that generalizations can form spontaneously. Second, generalizations are formed when the organism is trying to characterize a category. For example, in language learning, the use of the same word for two objects invites their comparison [Namy & Gentner, 2002], and during conceptual change, one might be trying to understand, for instance, the distinction between floating and sinking [Friedman & Forbus, 2008].

Our model of generalization is SEQL [Kuehne *et al.*, 2000]. SEQL takes as input a stream of examples. It maintains two lists, a list of generalizations and a list of exemplars. Given a new example, if it is sufficiently similar to one of the generalizations, as measured by SME's structural evaluation score being over a threshold, it is assimilated into it. If not assimilated into an existing generalization, it is compared with each example in the exemplar list, and if similar enough to one of them, they form a new generalization. Otherwise, the new example is added to the exemplar list. The assimilation process merges the corresponding facts together, and maintains probabilities for each fact in the generalization based on frequency of occurrence. In other research, we have shown that SEQL can be used to generate probabilistic rules [Halstead & Forbus, 2007] and to model the construction of causal knowledge during conceptual change [Friedman & Forbus, 2008].

Unlike the other two processes, SEQL is only now being integrated into Companions. We plan on using it in two ways: (1) Generalizations will be added to case libraries, and used in analogical reasoning just as examples are. (2) Companions will construct probabilistic rules for encoding that will be used in new situations. We believe that these additions will significantly enhance the power of the architecture.

Extensive Conceptual Knowledge

A hallmark of natural intelligent systems (e.g., humans) is that they know a lot. Consequently, a Companion is not an empty architectural shell nor a special-purpose problem-solver, but a general knowledge-rich agent that can acquire or learn domain knowledge by building on an extensive pre-existing ontology. The symbolic, relational structures required for expressing explanations, arguments, and plans are a key component of human mental life. We use the contents of the ResearchCyc knowledge base[1] as the starting point for the ontology. This enables a Companion to construct representations for many different domains (e.g. military planning, physics problems, and computer games).

One of the methodological problems that has vexed cognitive modelers is that, given the state of the art, it is

[1] ResearchCyc is described at www.research.cyc.com; for a historical overview, see [Lenat 1995].

simply impossible to psychologically vet anything like a large-scale knowledge representation system. One approach to that is to avoid modeling conceptual knowledge altogether. To us, that approach throws out the baby with the bathwater. Our approach is to treat ResearchCyc as an engineering approximation for human conceptual knowledge. Our experience to date is that it is quite satisfactory for that purpose.

While analogical reasoning is central to Companions, other kinds of reasoning, especially logical inference, are required. For example, logical inference is often used in checking the consistency and/or plausibility of candidate inferences, combining the results of multiple analogical inferences, bridging between queries and what is available from an analogy, and in dynamic case construction. These more limited roles for logical reasoning put less stress on it than an architecture that is completely rule-based, but many of the same problems of doing logical inference at scale still arise.

Large knowledge bases (KBs) present a pair of challenges for reasoning. (1) Large KBs are never complete or consistent. This, too, is a property of human knowledge. It is well documented in the mental models literature (e.g. [Gentner & Stevens, 1983]) that novices typically have multiple inconsistent models of a domain, for example. Furthermore, any well-trained scientist or engineer has multiple inconsistent models of the same phenomena, each useful under different circumstances (e.g., Newtonian versus relativistic dynamics). We have found that treating the microtheory structure of the KB as a specification of a logical environment for reasoning to be a workable approach to this problem [Lenat, 1995]. (2) Brute-force problem-solving methods suffer from prohibitively large search spaces as the number of facts and rules grows. Consequently, reflective control over reasoning becomes increasingly important. Companions address this in three ways. First, we use the usual resource limitations on search depth and elapsed time. Second, we do *partitioned reasoning*: We are never reasoning with the entire knowledge base, only a subset of relevant axioms as determined by the current logical environment. Third, we have a layered reasoning architecture that makes high-level reasoning decisions reflectively. In other words, logical reasoning is kept tightly constrained, sacrificing completeness for efficiency.

Flexible Federated Reasoning

A notable characteristic of human reasoning is what Simon referred to as bounded rationality [Simon, 83]. Models of reasoning cannot escape from the constraints imposed by the limited resources available to the human mind. Such bounded rationality is implicit in the design of FIRE, the Federated Integrated Reasoning Engine, which is the core inference engine of a Companion. FIRE is built on top of a Logic-based Truth Maintenance System (LTMS) which serves as a working memory cache and provides an audit trail for justifying and explaining inferences [Forbus & de Kleer, 1993]. Inference in FIRE is controlled by stratifying operations into fast low-level retrieval and local operations, constrained backchaining queries, and reflective queries that invoke And-Or problem-solving and HTN planning. All of these operations contain resource limits, ensuring that the Companion is responsive to user's requests.

FIRE achieves flexibility through federated operation. Rather than supporting arbitrary escapes to code in the middle of rules, it permits reasoning predicates to be defined procedurally ("outsourced") so that external algorithms, accessors and packages can be invoked. The answers of such external operations are packaged into declarative assertions that are fully justified in the LTMS, making these external systems transparent to the knowledge level aspects of the system. For example, analogical matching and retrieval are implemented as outsourced predicates, as are many spatial reasoning operations.

We are not making strong specific psychological claims about either an LTMS or the particular other reasoning mechanisms used in FIRE. That people are capable of some degree of reasoning than can be expressed via logic, that we have some ability to attribute reasons for our beliefs, and that we have some ways of generating plans and solving complex problems do seem to be psychologically plausible assumptions. The specific mechanisms we are using in this part of the system are more a function of engineering issues and convenience than theoretical commitments. The drawback, of course, is that this architecture cannot be used for generating low-level predictions about, for example, the exact timing of inference steps. On the other hand, our unit of modeling remains focused at conceptual reasoning and learning, and we do believe that these mechanisms are sufficient for modeling at that level.

Coarse-Grained Parallelism

Another design goal of Companions is to emulate the parallelism that is evident in human and animal behavior. Beyond simple multi-tasking, a large body of work in psychodynamics and personality theory suggests that the mind contains drives, emotions and instincts that conflict and compete for resources and attention [Rietman, 1963]. Such processes become increasingly important as we model bounded rationality in interactive systems, where inference, memory access, and communication with the user are time-limited resources.

Companions are implemented as distributed systems that allocate individual nodes of a cluster computer to semi-independent, asynchronous processes (agents). We use a small number of such agents per Companion, making this an example of *coarse-grained* parallelism. Agents communicate internally using KQML [Labrou *et al.*, 1997] with callbacks to support asynchronous queries and subscriptions to events. This interrupt-style invocation enables more bottom-up and heterogeneous control strategies that will allow us to experiment with some of the sorts of non-rational processes mentioned above.

A common criticism of coarse-grained parallelism is that fine-grain parallelism can often yield greater efficiency. This misses a critical point. Fine-grained parallelism is almost always applied to SIMD-type problems, where computations are homogeneous and repetitive. This is largely because it is prohibitive to manually program vast numbers of independent agents. Coarse-grained parallelism allows another sort of model, where sub-processes are fundamentally different, making them more independent and active.

Companions make use of a number of sub-processes. For example, human memory is often proactive, suggesting similar episodes and concepts that may be relevant. In Companions, the *Analogical Tickler* is an agent that effectively watches the state of working memory and continually retrieves cases to present to the user and the Session Reasoner (the agent responsible for domain reasoning). The Tickler operates on a subscription basis, so that a reasoner can request examples for, e.g., case-based reasoning. Another example of how we use coarse-grained parallelism is the *Executive* agent. The Executive is responsible for prioritizing work on the Companion's goals. For example, in learning to play games, it keeps an eye on the Session Reasoner, and "pulls the plug" if its learning does not seem to be converging.

Coarse-grained parallelism is also used in other architectures, such as PolyScheme [Cassimatis, 2006]. In PolyScheme, different reasoning mechanisms are invoked in parallel and race to produce an answer. By contrast, in a Companion, agents are functionally differentiated and dedicated to different tasks.

Ubiquitous Learning

People learn continually under all sorts of situations. Computers, however, typically learn only when directed to, and allocate all their resources to the task. This tends to be incompatible with highly interactive systems.

Companions address this in two ways: (1) compute-intensive learning tasks are off-loaded to background tasks on dedicated nodes, and (2) learning is focused via explicit learning goals that are constructed on the fly, prioritized, scheduled, and reasoned about. It is the job of the Executive agent to decide which learning goals should be pursued and how. We have used learning goals to drive learning in game domains, where plans were available to drive experimentation. We are in the process of adapting this approach to more general tasks, where the learning strategies might entail searching the KB for possible examples and counter-examples, or in some cases, simply asking the user.

Extended Lifetime

Natural intelligent systems exist continuously over their entire lifetime. They are not rebooted or shut down. Although people may sleep at night, they do not forget everything in the morning and their minds continue to operate in some fashion even when asleep. Consequently, a goal for Companions is to support extended interactive sessions and continuous operation between sessions.

Increasing the duration of sessions runs into the problem of hard resource limits on agents. It is not uncommon for the LTMS cache to fill up with facts that are no longer useful, until it eventually runs out of heap space and crashes the agent. To ameliorate this, Companions have the ability to query their own available heap, the number of TMS nodes allocated, and the number of reified analogical matches. An agent can choose to clear its cache and invoke garbage collection, but we do not yet have a good theory of when that is appropriate. More difficult is the problem of recognizing when an agent is about to crash and hot-swapping a fresh agent in its place. Again, while the low level actions are implemented, we are still working out strategies for when this should happen, ideally based on the system monitoring its own performance. These problems are very similar to the goals of IBM's autonomic computing initiative [Klephart & Chess, 2003].

One of the key benefits of supporting extended lifetimes is that a Companion can pursue compute-intensive learning tasks between sessions, and thereby apply learning to a greater variety of problems than would be possible with only online learning. We often refer to this as "homework" between sessions.

A second anticipated benefit is that as a Companion is applied to different domains over time, it will be able to apply analogy across domains to reuse strategies and build new abstractions. We have performed some initial experiments with cross-domain analogies but this is not yet a robust capability.

A third benefit of an extended lifetime is that it should enable the incremental construction of user and self-models. This is part of the ubiquitous learning goal, but it is only possible if a Companion remembers the context of prior sessions as well as the domain-specific problem solving. So far, Companions have built up case libraries of domain-specific episodes, in particular traces of problem solving and execution in game domains such as Freeciv and General Game Playing. As we begin to interact more via language, we expect to retain and learn more from linguistic interaction, possibly including, for example, resolutions of ambiguous parses. Self models may include, among other things, histories of resource usage to facilitate predicting when an agent should be restarted or hot-swapped.

Natural interaction modalities

Natural intelligent systems are able to interact robustly over a broad range of situations. Indeed, such interactions are crucial for learning in many domains. Our goal is for Companions to learn in Vygotskian fashion, while working as apprentices to their human partners. In addition to this lofty goal, there are also some quite practical reasons why natural interaction is crucial for a knowledge-rich agent. The ResearchCyc KB already has over 50,000 collections and several million facts in it. It would not surprise us for this number to double or triple as we learn how to achieve ubiquitous learning and extended lifetime in Companions. Even in a static knowledge base, knowledge engineers will define their own concepts if they cannot quickly find what they are looking for in the ontology [Cohen *et al.,* 1999]. When a significant fraction of what the system knows is automatically learned, manual inspection of internal representations will be hopeless as the only way of interacting. Thus supporting natural interaction is both essential given our theoretical commitments, and a practical matter as well.

We are currently experimenting with two modalities: natural language input and sketch-based interaction. A dedicated agent (Interaction Manager) is under development to coordinate interaction with the user. It invokes natural language parsing and semantic interpretation to translate English sentences into queries, statements, and commands[2]. The goal is to support learning through natural language tutoring, by, in effect, inverting the methods used in ICAI systems [Stevens *et al*., 1977][Kim & Gil, 2003]. Conceptual gaps or inconsistencies revealed during tutoring spawn learning goals that can be passed to the Executive for asynchronous or offline processing. Sketch-based interaction is supported by agents that encapsulate nuSketch applications and translate visual depictions into predicate calculus representations [Forbus *et al.*, 2004]. Sketch interaction has been already used in Companions for some early work on Tactical Decision Games and on mechanical comprehension tests [Klenk *et al.*, 2005]. In the latter project, the Companion would interact with the user by drawing on a sketch to explain its reasoning.

Experience with Companion Systems

Over the last few years, we have worked with Companions in a variety of different domains, including interactive games and question answering tasks in more content-focused domains.

AP Physics

We have applied Companions to the domain of Advanced Placement Physics tests. The AP Physics exam tests the ability of high-school students to solve physics problems. In collaboration with Cycorp and the Educational Testing Service (ETS), which administers the AP Physics exam, we evaluated a Companion's ability to solve problems of the style that would be found on the AP Physics exam. The Companion learned by accumulating examples and used these examples to make modeling decisions about new problems via analogy. For example, the Companion started out with basic algebra solving and problem decomposition skills, but zero knowledge of the equations of physics. All equations used in solving problems were found via analogies with worked solutions it had been given after prior exams, which were at the level of what might be found in a textbook. To address the ability to transfer knowledge from these examples to new situations, the evaluation centered around six systematic variations of problems, or transfer levels. These ranged from changing numerical parameters but not qualitative outcomes to requiring the composition of aspects of multiple examples.

In an external evaluation carried out by ETS, a Companion exhibited an average of 63.8% improvement in initial performance across all six transfer levels [Klenk & Forbus, 2007]. Interestingly, the sources of the problem-solving failures were primarily due to representation errors and some domain-specific strategies, not in our use of analogy. To verify this, we repeated the experiment after fixing these problems, and the Companion achieved an average of 98.5% improvement due to transfer.

Our experience in AP Physics problem-solving reinforces a number of our design decisions. First, the AP Physics problems are described in everyday terms requiring a large ontology. The 460 predicate calculus problem representations used in the evaluation included 108 conceptual types and 103 unique relations. Second, the Companion used analogy to make all of the decisions about which equations to use and what assumptions to make during each problem-solving episode. The Companion's strong performance demonstrates that analogy can play a central role in a complex reasoning task such as physics problem-solving.

Freeciv and General Game Playing

Using Companions to learn to play turn-based strategy games poses somewhat different challenges than taking tests. First, these domains require interleaving planning and action in a simulated environment. Second, games typically require reasoning with incomplete information. The Companion must make decisions, take actions, and evaluate outcomes. These factors led to the approach of representing explicit learning goals and having the agent formulate experiments to try in the simulated environment.

Freeciv[3] is an open source turn-based strategy game modeled after Sid Meier's series of Civilization^{TM} games. In the context of the DARPA Transfer Learning program, a Companion used analogy, experimentation, and qualitative modeling to improve performance in optimizing food production [Hinrichs *et al.*, 2007]. The Companion uses analogy to suggest worker allocations based upon successful previous cases. When this fails, the Companion uses experimentation to bootstrap the case library ensuring a variety of cases to reason from. The qualitative model allows the Companion to determine how the changes caused by an action affect the Companion's goals. This credit assignment is used to label precedents created by experimentation for future analogical reasoning episodes.

General Game Playing (GGP) [Genesereth & Love, 2005] is a framework for describing simple games declaratively such that the rules and premises of games can

[2] We are using the EA NLU system [Kuehne & Forbus, 2004] as the starting point.

[3] www.freeciv.com

be easily modified to exercise flexible reasoning and transfer of learning. We applied Companions to the task of transfer learning strategies for winning 2D board games in GGP. This continued the learning goal approach that began with the Freeciv work, but relied even more heavily on the analogy mechanisms in order to map across increasingly distant base-target pairs. Coupled with the experimentation strategy, we were able to learn new HTN plans for games that were structurally different from games learned previously.

Our work in these domains provides evidence for the importance of the design decisions presented in this overview. First, Companions reasoning in these domains exhibited the beginnings of ubiquitous learning. Driving experimentation via learning goals proved to be an effective method for learning about these domains. In the future, we plan on making this more reflective, empowering the Companion to spawn new agents with specific learning goals. Also, analogy was central to reasoning about these domains. Companions used analogy to suggest individual actions in new situations, find commonalities between domains, and transfer entire plans.

Challenges

Our development of Companions has frequently run into challenges of two types: engineering and evaluation. From an engineering standpoint, achieving these design features in Companions poses a number of difficulties. One challenge concerns adapting Companions to new domains. While the underlying KB allows representations for a wide range of domains quickly, there is still the problem of interfacing the Companion with its environment. In the game playing domain, it was necessary to build agents to interface between the Companion and the game programs.

Another engineering challenge concerns the size of the knowledge base. While our design decisions seek to mitigate this, there are still important engineering decisions concerning efficiency and coordination between the agents. To address efficiency concerns, the FIRE reasoning engine has recently been revised with a new backend knowledge base, built on the Franz AllegroCache[4] persistent object store. As agents within the Companion learn, it is necessary that the changes to the individual knowledge bases are synchronized across the system. We use a journaling mechanism, in which KB changes are stored and shared with the other agents.

Evaluation of intelligent architectures is difficult. In fact, it has recently received increased attention from AI researchers [Kaminka & Burghart 2007]. A major frustration we have had is that the metrics adopted for evaluating learning, a hallmark of intelligent systems, has tended to favor batch over interactive operation. This is a major problem because three of the design features, natural interaction, extended lifetime, and ubiquitous learning, are important in interactive systems. It is fundamentally difficult to evaluate interactive systems, and this has hindered progress on many of the more interesting aspects of Companions. We suspect that the way around this is to focus on measuring relative rates of learning, rather than on absolute measures.

[4] http://www.franz.com/products/allegrocache/

Discussion

While the Companions architecture is a work in progress, we have already had a number of successes. First, we have used Companions for reasoning in a wide variety of domains: everyday physical reasoning, tactical decision games, Freeciv city planning, AP Physics problem-solving and general game learning. Second, Companions have been run, interacted with, and evaluated externally by two collaborators: ETS and the Naval Research Laboratory (NRL). Our experiences reinforce our commitment to the design decisions that are shaping Companions: (1) the centrality of analogy, (2) extensive conceptual knowledge, (3) flexible federated reasoning, (4) coarse-grained parallelism, (5) ubiquitous learning, (6) extended lifetime, and (7) natural interaction modalities. As this article illustrates, human behavior suggests many frontiers on which to push artificial intelligence models. Companions is an attempt to explore parts of that space that we think are essential to achieving human-level artificial intelligence, but have not been explored much yet.

Acknowledgements

This work has been supported by the Defense Advanced Research Projects Agency under an IPTO seedling and the Transfer Learning Program, and by the Office of Naval Research, under the Intelligent Systems and Cognitive Science Programs.

References

Anderson & Lebiere (1998). *The Atomic Components of Thought*. Erlbaum.

Cassimatis, N. 2006. A Cognitive Substrate for Human-Level Intelligence. *AI Magazine,* vol. 27(2):45-56.

Cohen, P., V. Chaudrhi, A. Pease, and R. Schrag, 1999. Does Prior Knowledge Facilitate the Development of Knowledge-based Systems? *Proceedings of the 16th National Conference on Artificial Intelligence*.

Ellman, T., 1989. Explanation-based learning: a survey of programs and perspectives. *ACM Computing Surveys (CSUR)*. 21(2).

Falkenhainer, B., Forbus, K., and Gentner, D. 1989. The Structure-Mapping Engine: Algorithm and Examples. *Artificial Intelligence* 41(1):1-63.

Forbus, K. (2001). Exploring analogy in the large. In Gentner, D., Holyoak, K., and Kokinov, B. (Eds.) *Analogy: Perspectives from Cognitive Science*. MIT Press.

Forbus, K. and de Kleer, J. 1993. *Building Problem Solvers*. MIT Press.

Forbus, K., Gentner, D., and Law, K. (1994). MAC/FAC: A model of similarity-based retrieval. *Cognitive Science*, 19, 141-205.

Friedman, S., and Forbus, K. 2008. Learning Causal Models via Progressive Alignment & Qualitative Modeling: A Simulation. In *the Proceedings of CogSci-08*. Washington, D.C.

Genesereth, M., and Love, M. 2005. General game playing: Overview of the AAAI competition. *AI Magazine*, 26(2).

Gentner, D. 1983. Structure-Mapping: A theoretical framework for analogy, *Cognitive Science* 7(2):155-170.

Gentner, D. and Stevens, A. 1983. *Mental Models*. Erlbaum.

Gentner, D. 2003. Why we're so smart. In *Language in Mind* (Gentner, D. and Goldin-Meadow, S., eds) MIT Press.

Halstead, D. and Forbus, K. 2007. Some Effects of a Reduced Relational Vocabulary on the Whodunit Problem. *Proceedings of IJCAI-2007*, Hyderabad, India.

Hinrichs, T. and Forbus, K. 2007. Analogical Learning in a Turn-Based Strategy Game. *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, 853-858.

Kaminka, G. and Burghart, C. (Eds.) 2007. Evaluating Architectures for Intelligence. *Papers from the 2007 AAAI Workshop.* Technical Report WS-07-04, AAAI Press.

Kephart, J. and Chess, D. 2003. The Vision of Autonomic Computing. *IEEE Computer Magazine*, January: 41- 50.

Kim, J. and Gil, Y. 2003. Proactive acquisition from tutoring and learning principles. *Proceeding of AIEd2003*, Sidney, Australia.

Klenk, M., Forbus, K., Tomai, E., Kim, H., and Kyckelhahn, B. 2005. Solving Everyday Physical Reasoning Problems by Analogy using Sketches. *Proceedings of 20th National Conference on Artificial Intelligence*, Pittsburgh, PA.

Klenk, M. and Forbus, K. 2007. Measuring the level of transfer learning by an AP Physics problem-solver. *Proceedings of AAAI-07: Twenty-Second Conference on Artificial Intelligence*, Vancouver, BC.

Kuehne, S. and Forbus, K. 2004. Capturing QP-relevant information from natural language text. *Proceedings of QR2004*.

Kuehne, S., Forbus, K., Gentner, D. and Quinn, B. 2000. SEQL: Category learning as progressive abstraction using structure mapping. *Proceedings of Annual Conference of the Cognitive Science Society.*

Labrou, Y. and Finin, T. 1997. A proposal for a new KQML specification. Technical Report TR CS-97-03, University of Maryland Baltimore County.

Laird, J. 2008. Extending the Soar Cognitive Architecture. *Artificial General Intelligence Conference*, Memphis, TN.

Langley, P. and Choi, D. (2005) A unified cognitive architecture for physical agents. *Proceedings of AAAI05.*

Lenat, D. 1995. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM* 38(11):33–38.

Namy, L., and Gentner, D. 2002. Making a silk purse out of two sow's ears: Young children's use of comparison in category learning. *Journal of Experimental Psychology:General* 131(1):5-15

Rietman, W. 1963. Personality as a Problem-Solving Coalition. In *Computer Simulation of Personality* (Tomkins and Messick, eds.). John Wiley & Sons.

Ross, B. 1989. Distinguishing types of superficial similarities: Different effects on the access and use of earlier examples. *Journal of Experimental Psychology: Learning, Memory, and Cognition*. 15(3),456-468.

Simon, H. 1983. Reason in Human Affairs, Stanford University Press.

Stevens, A. and Collins, A. 1977. The Goal Structure of a Socratic Tutor. In *Proceedings of the ACM National Conference.* pp .256-263.

Tomasello, M. (1999). *The Cultural Origins of Human Cognition*. Harvard University Press.

Vygotsky, L. (1962) *Thought and Language*. MIT Press.

Wahlster, W. 2000. *Verbmobil: Foundations of Speech-to-Speech Translation.* Springer.