

Companion Cognitive Systems: Design Goals and Lessons Learned So Far

Kenneth D. Forbus, Matthew Klenk, and Thomas Hinrichs, *Northwestern University*

Human beings are organisms. They interact with each other and their environment to perform a variety of tasks over a broad period of time.

This observation is obvious, but surprisingly, it doesn't seem to play a central role in most current cognitive architectures. Perhaps even more important,

The Companion cognitive architecture supports experiments in achieving human-level intelligence. Its design goals relate to human reasoning, learning, and engineering large-scale cognitive systems.

all natural intelligent systems we know of are social organisms. Indeed, some speculate that social behavior, not manipulation or tool making, has been the major driving force in the evolution of intelligence. A fundamental goal of the Companion cognitive architecture is understanding how to build intelligent systems that are social beings. Lev Vygotsky argued that much of our knowledge is learned from interactions with others in our culture, with apprenticeship being especially important.¹ We believe we can use the same approach to create cognitive systems that operate more closely to human-level intelligence. We consider Companions as the first attempt to create a cognitive architecture that includes Vygotskian influences.

This is clearly a very different point in the space of possible cognitive architectures, compared to skill-oriented architectures such as ACT-R,² Soar,³ PolyScheme,⁴ Icarus,⁵ and others. Another significant difference is that we've been motivated by the

growing body of evidence that analogical processing is a core operation of human cognition. Dedre Gentner and her colleagues have been amassing evidence that structure-mapping operations occur everywhere from medium-level vision up through conceptual change.⁶ Companions take analogical processing, as defined by structure-mapping theory (see the "Structure-Mapping Theory and Simulations" sidebar), as fundamental to their operation.

Central Design Goals

Every implemented cognitive architecture has design decisions based on theoretical bets and empirical evidence, and others based on engineering concerns. There are seven key features of our current design: analogical processing, extensive conceptual knowledge, flexible reasoning, coarse-grained distributed implementation, ubiquitous learning at multiple levels, long-lived continuous operation, and natural interaction. Some of these design decisions have

Structure-Mapping Theory and Simulations

The Companions architecture uses Dedre Gentner's *structure-mapping theory* of analogy and similarity. In structure-mapping, analogy and similarity involve comparisons between structured representations.¹ This is in sharp contrast to prior psychological models, which relied on feature vectors or multidimensional spaces. In comparing two descriptions, a *base* and a *target*, one or more mappings are generated. Here the base is the top description, with entities depicted by squares, predicates by ovals, and argument links by arrows (see Figure A).

Mappings consist of *correspondences* indicating what items in the base go with what items in the target (the red dotted lines); a *structural evaluation score* providing an estimate of match quality; and *candidate inferences* (in blue). These inferences are conjectures about the target constructed by projecting facts from the base that are only partially mapped.

Considerable psychological evidence now supports structure-mapping theory. Furthermore, researchers have used simulations of key processes postulated by the theory (comparison, similarity-based retrieval, and analogical generalization) to model a variety of phenomena and make novel predictions. These simulations serve as key components in the Companion architecture. This building up of models, layer by layer, represents a new direction for cognitive simulation.

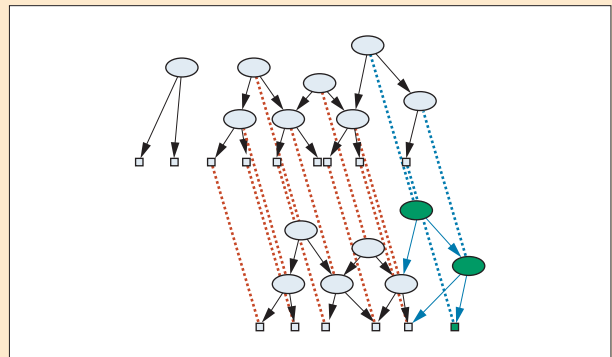


Figure A. An example of an analogical match. Analogies involve finding correspondences between structured representations. This includes projecting information from one description to the other, providing a form of pattern completion.

Reference

1. D. Gentner, "Structure-Mapping: A Theoretical Framework for Analogy," *Cognitive Science*, vol. 7, no. 2, 1983, pp. 155–170.

solid theoretical foundations and are well-tested, while others are still in flux.

The Centrality of Analogy

A major hypothesis of the Companion cognitive architecture is that analogical processing is central to human reasoning and learning.⁷ This is a radical difference from other cognitive architectures, where analogy is generally ignored. In the Companion architecture, analogy plays three major roles: matching, memory retrieval, and generalization.

The traditional hallmark of analogical matching is distant, cross-domain mappings, such as understanding that heat is like water. Although such cross-domain mappings are indeed important, they're far from the whole story. For example, when you learn that you can start a car by using a key, 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 on comparisons between two structured representations.

Our model for analogical matching is the Structure-Mapping Engine (SME).⁶ It computes mappings using a greedy algorithm, operating in polynomial time. Exploiting analogies is central to Companion 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 (for instance, examples in textbooks and morals in fables). Analogical reasoning in the "inner loop" means that you 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 that

can easily lead to combinatorial explosions in broad domains. Importing a whole relational structure is like striding through an inference space with seven-league boots.

Similarity-based retrieval finds potentially useful prior experiences. Psychological evidence has consistently shown that human memory retrieval is sensitive to surface information, not just deep relational structure. This makes sense if you consider that most analogies made in an organism's daily life are within-domain comparisons. Spontaneous cross-domain retrievals are relatively rare, unless there's 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.

Our model for similarity-based retrieval is MAC/FAC (Many Are Called/Few Are Chosen).⁸ MAC/FAC

takes as input a *probe* and a *case library*. It returns one or more cases from the library that best approximate the probe. In a Companion, the probe is typically the contents of working memory. A comparison with standard case-based reasoning (CBR) retrieval systems provides a useful contrast. The majority of CBR systems today only use feature vector representations, which means they can't 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 doesn't require any hand-indexing. The first stage (MAC) uses a special kind of feature vector, automatically constructed from relational representations, whose dot product provides an estimate of how strong the SME match would be on the relational representations. The output of this highly parallel first stage is fed into the FAC stage, which then runs SME in parallel on the relational form of the candidates found by the MAC stage. At most only three candidates can be returned from MAC, while memory can consist of (potentially) millions of cases—hence, the system's name. In Companions, the combination of analogical retrieval and matching has provided a simple but 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. Because people often don't have complete and correct theories of the domains they deal with, such caution is wise. On the other hand, people clearly learn much faster than today's statistical learning algorithms

do. 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. Similarly, during conceptual change, someone might be trying to under-

The combination of analogical retrieval and matching has provided a powerful learning mechanism: learning by accumulating examples.

stand, for instance, the distinction between floating and sinking.

Our model for generalization is SEQL (Sequential Learning).⁹ SEQL takes as input a stream of examples. It maintains two lists: a list of generalizations and a list of exemplars. If a new example is sufficiently similar to one of the generalizations, as measured by SME's structural evaluation score being over a threshold, it's assimilated into that generalization. If it isn't assimilated into an existing generalization, it's compared with each example in the exemplar list. If it's similar enough to one of the exemplars, they form a new generalization. Otherwise, the new example is added to the exemplar list. The assimilation process merges the corresponding facts and maintains probabilities for each fact in the generalization based

on frequency of occurrence. These probabilities can be used in analogical reasoning directly, providing an estimate of how likely a candidate inference is. They can also be used to automatically generate abstract probabilistic rules or Bayesian networks from the generalization.

Unlike the other two analogical mechanisms, we're only now integrating SEQL into Companions. We plan on using it in two ways. First, generalizations will be included in case libraries, so that Companions will use them in analogical reasoning just as examples are. Second, Companions will construct probabilistic rules for encoding—that is, rules that elaborate a situation description based on what experience indicates is worth noticing. We believe that these additions will significantly enhance the power of the architecture.

The only other cognitive architecture that has attempted to model analogy at all is ACT-R.¹⁰ Its model of analogical mapping requires externally specified values for chunk similarity for each pair of chunks, plus representations where what can be aligned has already been specified a priori. In contrast, SME's structural-alignment process dynamically computes what entities and statements can align, and automatically calculates a numerical similarity of descriptions in ways that accord well with human data. ACT-R's model of memory involves recency and utility estimates, properties that MAC/FAC currently ignores. Although they might be needed as we scale up, our similarity metric has proven extremely useful, as the following experiments indicate.

Extensive Conceptual Knowledge

Human-level intelligence depends on extensive knowledge about the world. Consequently, a Companion is nei-

ther 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 preexisting ontology. The symbolic, relational structures required for expressing episodes, explanations, arguments, and plans are a key component of human mental life. We use content extracted from the ResearchCyc knowledge base (<http://research.cyc.com>) as the starting point for the ontology. This enables a Companion to construct representations for many different domains (for example, military planning, physics problems, and computer games).

One of the methodological problems that's vexed cognitive modelers is that, given the state of the art, it's simply impossible to psychologically vet anything like a large-scale knowledge representation system. One common solution is to avoid modeling conceptual knowledge altogether. To us, that approach throws out the baby with the bathwater. Our approach is to treat ResearchCyc knowledge as an engineering approximation for human conceptual knowledge. Our experience to date is that it's quite satisfactory for that purpose.

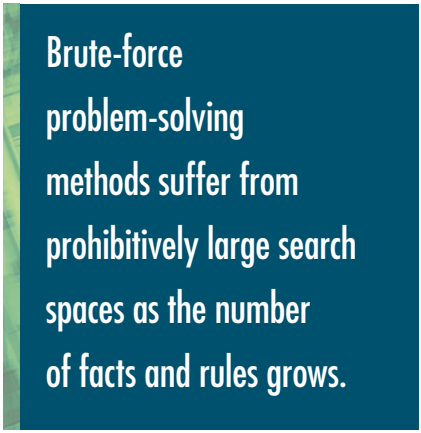
Although analogical reasoning is central to Companions, other kinds of reasoning, especially logical inference, are required. For example, logical inference is often used in

- checking the consistency and/or plausibility of candidate inferences,
- combining the results of multiple analogical inferences,
- bridging between queries and what's available from an analogy, and
- 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. First, large KBs are never complete or consistent. This, too, is a property of human knowledge. For example, it's well documented in the mental-models literature that novices typically have multiple inconsistent mod-



Brute-force
problem-solving
methods suffer from
prohibitively large search
spaces as the number
of facts and rules grows.

els of a domain.¹¹ Furthermore, any well-trained scientist or engineer has multiple inconsistent models of the same phenomena, each useful under different circumstances (for example, Newtonian versus relativistic dynamics). We have found that treating the microtheory structure of the KB as a specification of a logical environment for reasoning is a workable approach to this problem.

The second challenge is that 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. In Companions we address this in three ways. First, we use resource limitations on search depth and elapsed time. Second, we partition reasoning such that the

system never reasons with the entire knowledge base but with 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. As we further explain in the next section, logical reasoning is kept tightly constrained, sacrificing completeness for efficiency.

Flexible Federated Reasoning

A notable characteristic of human reasoning is what Herbert Simon referred to as *bounded rationality*.¹² That is, models of reasoning can't escape from the constraints imposed by the limited resources available to the human mind. Such bounded rationality is implicit in the design of the Federated Integrated Reasoning Engine (FIRE), which is the core inference engine of a Companion. FIRE is built on top of a Logic-based Truth Maintenance System (LTMS) that serves as a working memory cache and provides an audit trail for justifying and explaining inferences. 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 hierarchical task network (HTN) planning. All of these operations contain resource limits, helping to ensure that Companions are responsive to their users' requests.

FIRE achieves flexibility through federated operation. Rather than supporting arbitrary escapes to code in the middle of rules, FIRE provides an "outsourcing" interface, allowing implementers to define reasoning predicates in terms of calls to external algorithms and software packages. This interface packages the answers of such external operations into declarative assertions that are fully

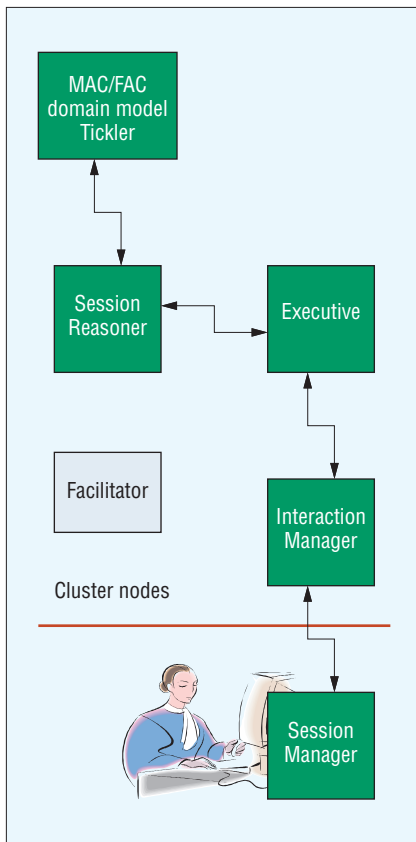


Figure 1. Agents in a typical Companion. Most of the agents in a Companion reside on a cluster. Reasoning about the domain, the user, and itself are handled by separate agents (Session Reasoner, Interaction Manager, and Executive, respectively). Because analogy plays a central role, retrieval is handled by a separate agent (Tickler).

justified in the LTMS, making these external systems transparent to the knowledge-level aspects of the system. For example, in FIRE analogical matching and retrieval are implemented as outsourced predicates, as are many spatial-reasoning operations.

We aren't making psychological claims about either the LTMS or other nonanalogical reasoning mechanisms used in FIRE. We think it's psychologically plausible to assume that people are capable of doing some degree of logical reasoning, that we can often attribute reasons to beliefs we hold, and that we have ways of generating plans and solving complex problems.

The specific mechanisms we're using in the nonanalogical parts of the system are more a function of engineering solutions to these problems than theoretical commitments. The drawback, of course, is that this architecture can't be used for generating low-level predictions about, for example, the exact timing of inference steps, as ACT-R and Soar do. 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. For example, we have used Companions to model the analogy events that occur during human physics problem-solving.¹³

Coarse-Grained Parallelism

Another design goal for Companions is to emulate the parallelism that's evident in human behavior. Beyond simple multitasking, 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. 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 the Knowledge Query and Manipulation Language (KQML) 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 nonrational processes previously mentioned.

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

Companions use a number of subprocesses (see Figure 1). For example, human memory is often proactive, suggesting similar episodes and concepts that might 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 to the Session Reasoner, the agent responsible for domain reasoning. The Tickler operates on a subscription basis, so that another agent can request examples from a particular case library. Analogical comparison allows for many types of reasoning; one common use is for CBR. 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 doesn't seem to be converging.

Experimenters can start up Companions in different configurations, defined by the collection of agents used. For some domains, additional agents are used compared to the baseline configuration illustrated in Figure

1. When a task requires sketches, for example, an additional agent linked to the Session Manager handles the sketch-understanding process.

Other architectures, such as PolyScheme, also use coarse-grained parallelism. PolyScheme invokes different reasoning mechanisms in parallel that race to produce an answer. In contrast, in a Companion, agents are functionally differentiated and dedicated to different tasks.

Ubiquitous Learning

People learn continually in 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.

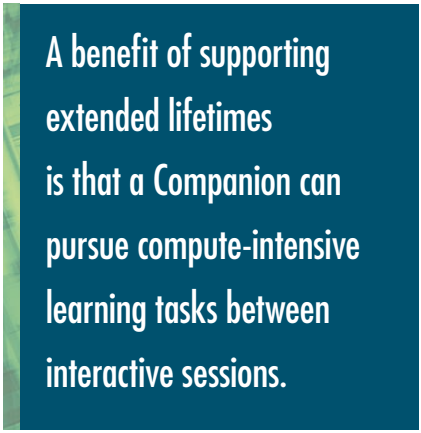
The Companion architecture addresses this in two ways:

- Compute-intensive learning tasks are off-loaded to background tasks on dedicated nodes.
- Learning is focused via explicit learning goals that are constructed on the fly, prioritized, scheduled, and reasoned about.

Part of the Executive agent's job is to decide which learning goals should be pursued and how. We've used learning goals to drive learning in game domains, where plans were available to drive experimentation. We're in the process of adapting this approach to more general tasks, where the learning strategies might entail searching the KB for possible examples and counterexamples or, in some cases, simply asking the user.

Another significant difference between AI learning algorithms and human learning regards the type of knowledge acquired. An AI learning system is typically designed with specific types of goal knowledge in mind. For example, can I learn rules

to categorize these examples in different classes? Or in this particular game, what action should I take in a given situation? When people learn a subject, they learn more than just the domain knowledge. They also learn about how knowledge is communicated in the domain and about their own expertise and understanding. Ubiquitous learning includes learning about the domain, learning about others, and learning about yourself.



A benefit of supporting extended lifetimes is that a Companion can pursue compute-intensive learning tasks between interactive sessions.

An important prerequisite to achieving this type of learning is the next design goal, extended lifetime.

Extended Lifetime

Human beings live for decades and exist continuously over their entire lifetime. They're not rebooted or shut down. Although people might sleep at night, they don't 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 interactive sessions.

Increasing the duration of sessions runs into the problem of hard resource limits on agents. It isn't 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 space, the number of LTMS nodes allocated, and the number of reified analogical matches. An agent can choose to clear its cache and invoke garbage collection, but we don't yet have a good theory of when that's 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, although the low-level actions are implemented, we're 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.

One of the key benefits of supporting extended lifetimes is that a Companion can pursue compute-intensive learning tasks between interactive sessions, and thereby apply learning to a greater variety of problems than would be possible with only on-line learning. We often refer to this as "homework" between interactive 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've performed some initial experiments with cross-domain analogies, but this capability hasn't yet been extensively tested.

A third benefit of an extended lifetime is that it should enable the incremental construction of user models and self-models. This is part of the ubiquitous-learning goal, but it's possible only 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 do-

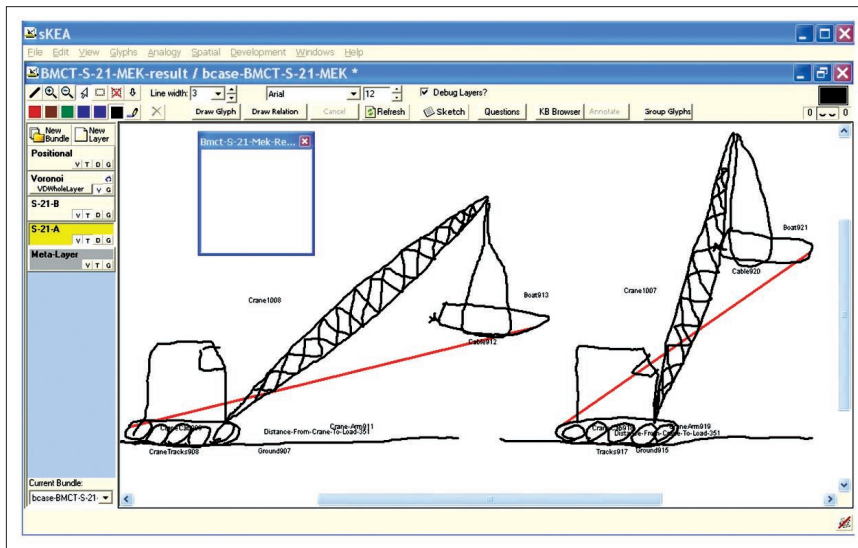


Figure 2. Crane problem. A Companion solves an everyday physics problem via causal and visual reasoning, on the basis of an analogy with prior experience. In determining which crane is more stable, it retrieves a prior example drawn by someone else and compares it to both of the new cranes. The explanation in the example, that stability goes up as the distance of the load to the cabin goes down, is used with a visual comparison between the two cranes to determine that the crane on the right is more stable.

main-specific episodes, in particular traces of problem-solving and execution in game domains such as Freeciv and General Game Playing (GGP), as discussed later. 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 might include, among other things, histories of resource usage to facilitate predicting when an agent should be restarted or hot-swapped.

Natural Interaction Modalities

Humans 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, 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 Research-Cyc KB already contains over 50,000 collections and several million facts. It

wouldn't 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 can't quickly find what they're looking for in the ontology. 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 essential, given our theoretical commitments and as a practical matter.

We're currently experimenting with two modalities: natural language input and sketch-based interaction. A dedicated agent, the *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. The goal is to support learning through natural language tutoring, by, in effect, inverting the methods used in intelligent tutor-

ing systems. 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 existing sketch-understanding software that translates visual depictions into predicate-calculus representations. We have already used sketch interaction in Companions for some early work on tactical decision games and on mechanical comprehension tests.¹⁴ In the latter project, the Companion and the user could interact by drawing on a sketch to explain its reasoning. For example, using a sketch, a user could explain that the stability of a crane is causally related to the distance between the base of the crane and its load. Then, in a later problem-solving episode, the Companion could determine which of two sketched cranes was more stable and illustrate for the user the relevant measurements that led to its answer (see Figure 2).

Experience with Companion Systems

Over the last few years, we've worked with Companions in a variety of different domains, including test-taking and interactive games.

AP Physics

The AP (advanced placement) Physics exam tests the ability of US high-school students to solve physics problems. It is administered by the Educational Testing Service (ETS). ETS, with assistance from Cycorp, evaluated a Companion's ability to learn to solve AP physics-style problems by analogical transfer (see Figure 3).

For this experiment, the Companion started out with basic algebra solving and problem decomposition skills but no knowledge of the equations of physics. Whenever it was

given a quiz, it would receive worked solutions afterward. The worked solutions were at the level of detail that's found in physics textbooks. Such worked solutions aren't proofs, because they leave out many details. To factor out natural language processing, both problems and worked solutions were provided as predicate calculus.

The Companion learned by accumulating pairs of problems and worked solutions. Given a new problem, it used the Tickler (running MAC/FAC) to retrieve a relevant analog, using the analogy to suggest equations and assumptions that might be used in solving the new problem.

To explore the ability to perform transfer learning, six systematic variations between base and target pairs were tested. These included small changes to numerical parameters (for example, changing the mass of an object by 10 percent), changes to numerical parameters large enough to cause qualitatively different outcomes (for example, dropping a ball off a house instead of a skyscraper), solving for a different parameter, adding distracters, using different types of objects, and composing multiple problem types. Unlike most analogy experiments, where the system designers also generate the problems used in testing, we saw only roughly 34 percent of the more than 400 problems involved in the ETS experiments in advance.

A Companion handled this challenge quite well, exhibiting an average of 63.8 percent improvement in initial performance across all six transfer conditions.¹⁵ 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 aver-

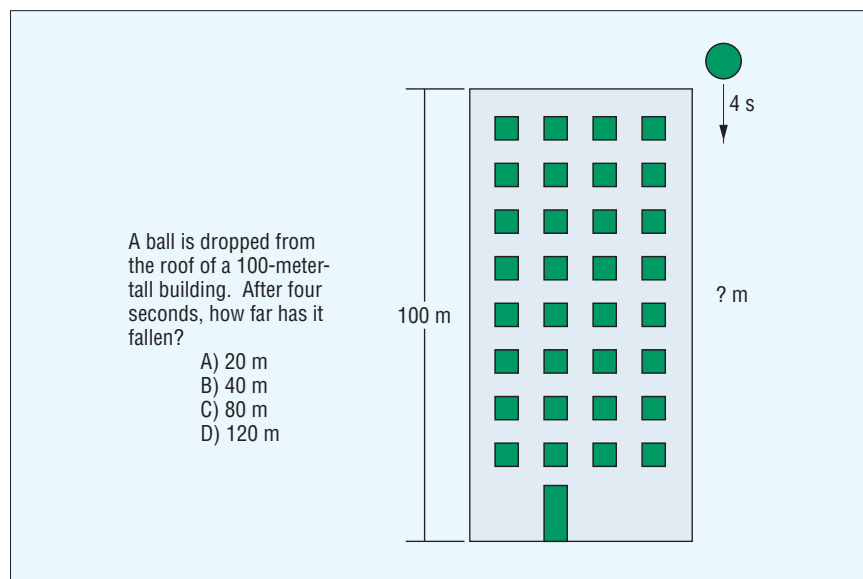


Figure 3. Example of an AP (Advanced Placement) Physics-style problem. Problems like these are given to US high school students to test their knowledge of physics. Companions were given predicate calculus versions of these problems, and learned to solve novel problems by analogy with worked solutions.

age of 98.5 percent 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 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. Moreover, given the starting conditions of no domain-specific equations or knowledge of how to make assumptions, no architecture could perform this task without analogical abilities.

Freeciv and GGP

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. A Companion must make decisions, take actions, and evaluate possible outcomes. These factors led to the approach of representing explicit learning goals and having the companion formulate experiments to try in the simulated environment.

Freeciv (www.freeciv.org) is an open source turn-based strategy game modeled after Sid Meier's series of Civilization 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.¹⁶ The Companion used analogy to suggest worker allocations based on successful previous cases. When this failed, the Companion used experimentation to generate a set of new cases, to bootstrap the case library and thus provide a variety of cases to reason from. A qualitative model allowed the Companion to determine how the

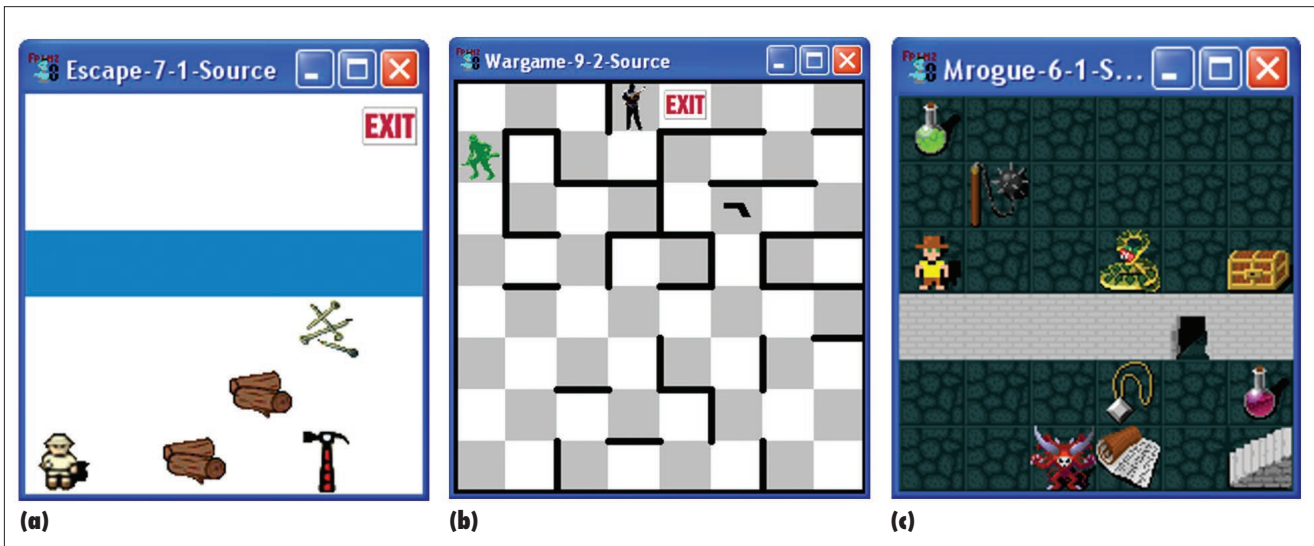


Figure 4. Families of games used in a transfer learning experiment. (a) *Escape* is a 2D board game where the explorer must reach an exit, by combining materials to overcome obstacles. (b) *Wargame* involves a soldier trying to reach an exit, finding weapons and killing terrorists along the way. (c) *mRogue* is a simplified version of the popular *Rogue* dungeon game.

changes caused by an action affected its goals. The Companion used this credit assignment to label precedents created by experimentation for future analogical-reasoning episodes.

GGP (<http://games.stanford.edu>) is a framework for describing simple games declaratively such that the rules and premises of games can be easily modified to exercise flexible reasoning and transfer of learning. In the same DARPA Transfer Learning project, we applied Companions to the task of transfer-learning strategies for winning 2D board games in GGP. Three families of games were used, as illustrated in Figure 4. *Escape*-style games require combining resources to overcome a barrier. The *Wargame* family of games has a soldier trying to reach an exit, killing a terrorist on his way. The *mRogue* family of games is based on the classic Internet game, *Rogue*, a role-playing game in which players collect armor, weapons, and treasures while killing monsters.

The GGP experiments continued the learning-goal approach that began with the *Freeciv* work. That is, the rules of the game (expressed via GGP datalog-style rules) were auto-

matically analyzed, identifying what actions were available and possibly relevant quantities. The Companion learned models of actions, tactics, and strategies via experimentation, driven by learning goals. The Executive managed this process, deciding when the Companion sufficiently understood a game to go on. We also used the Executive to orchestrate batch experiments, notifying us via email about the results.

For transfer learning, the Companion was given a previously learned game available as a starting point. This factored out retrieval as an issue. Because the Companion derived the bulk of the knowledge about each game via an automatic analysis process, almost none of the predicates were the same across two games, even when they were from the same family of game. This made them all into distant-transfer problems, which are much harder. Nevertheless, in an experiment conducted by the US Naval Research Laboratory, a Companion was able to learn 45 particular games from these three families via transfer.

Specifically, we measured the regret scores (see Figure 5) for learning

games with and without transfer. Regret is a normalized measure of the difference between the areas of the learning curves in the transfer versus nontransfer conditions. Positive regret is desired, because that means the transferred knowledge sped up learning. A Companion achieved positive transfer 59 percent of the time and negative transfer only 15 percent of the time. The average positive transfer was 59 percent, and the average negative transfer was -26 percent. In other words, most of the time transferred knowledge helped, and negative transfer was both much less likely and smaller when it occurred.

Our work in these domains provides evidence for the importance of the design decisions we presented here. First, Companion reasoning in these domains exhibited the beginnings of ubiquitous learning. Driving experimentation via learning goals proved an effective method for learning about these domains. In the future, we plan on making this process more reflective and empowering Companions 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. This external evaluation is the first we know of where the base/target pairs were not generated by the experimenters themselves. Moreover, the number of novel distant base/target pairs used in this experiment (41) is larger than in any other experiment we know of in the literature, where researchers typically use only two or three pairs.

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 difficulty concerns adapting Companions to new domains. Although the underlying KB supports constructing representations for a wide range of domains quickly, there's still the problem of interfacing a Companion with its environment. In the game-playing domain, it was necessary to build agents to interface between Companions and the game programs.

Another engineering difficulty concerns the size of the knowledge base. Although our design decisions seek to mitigate this, there are still important engineering decisions concerning efficiency and coordination between the agents. To address efficiency concerns, we recently revised the FIRE reasoning engine with a new back-end knowledge base, built on the Franz AllegroCache (www.franz.com/products/allegrocache) persistent object store. As agents within a Companion learn, it's 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.

Evaluating intelligent architectures is difficult. One frustration we've had is that the metrics adopted for evaluating learning, a hallmark of intelligent systems, have tended to favor batch processes over interactive operation. This is a major problem because three of our design goals—natural interaction, extended lifetime, and ubiquitous learning—are intrinsically interactive. It's fundamentally difficult to evaluate interactive systems, and this has hindered progress on many of the more interesting aspects of Companions.

Although the Companion architecture is a work in progress, we have already had a number of successes. First, we've used Companions for reasoning in a wide variety of domains, as we described in this article. Second, Companions have been run, interacted with, and evaluated externally by two collaborators: the Educational Testing Service and the Naval Research Laboratory. Our experiences reinforce our commitment to the design decisions that are shaping Companions: the centrality of analogy, extensive conceptual knowledge,

flexible federated reasoning, coarse-grained parallelism, ubiquitous learning, extended lifetime, and natural interaction modalities.

As this article illustrates, human behavior suggests many frontiers on which to push AI models. The Companion architecture is exploring ideas that we believe will ultimately prove essential in achieving human-level AI. Although the effort is still young, we're very excited and encouraged about the progress to date. ■

Acknowledgments

This work has been supported by DARPA under an Information Processing Techniques Office seedling and the Transfer Learning Program, and by the US Office of Naval Research under the Intelligent Systems and Cognitive Science Programs.

References

1. L. Vygotsky, *Thought and Language*, MIT Press, 1962.
2. J.R. Anderson and C.J. Lebiere, *The Atomic Components of Thought*, Lawrence Erlbaum Associates, 1998.
3. J. Laird, "Extending the Soar Cognitive Architecture," *Proc. 1st Ann. Artificial*

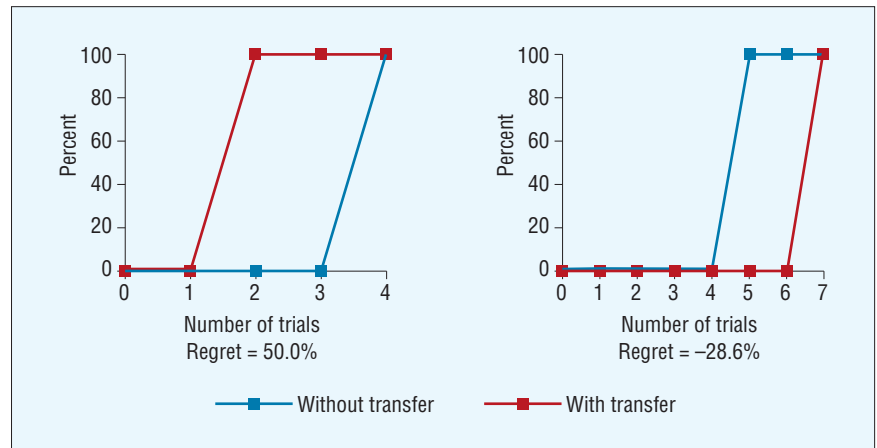


Figure 5. Examples of regret scores. The y-axis is the game score, 100 being the maximum. Regret is the normalized difference between areas under the learning curves. Positive regret means that transferred knowledge helped; negative regret means that it hurt.

THE AUTHORS

Kenneth D. Forbus is the Walter P. Murphy Professor of Computer Science and a professor of education at Northwestern University. His research interests include qualitative reasoning, analogy and similarity, sketch understanding, spatial reasoning, cognitive simulation, cognitive architecture, reasoning-system design, articulate educational software, and AI's roles in interactive entertainment. Forbus has a PhD in artificial intelligence from the Massachusetts Institute of Technology. He's a fellow of the AAAI, the Cognitive Science Society, and the ACM. Contact him at forbus@northwestern.edu.

Matthew Klenk is a National Research Council postdoctoral research associate at the US Naval Research Laboratory. He received his PhD in electrical engineering and computer science with a specialization in cognitive science from Northwestern University in June 2009. Klenk is a member of the AAAI and the Cognitive Science Society. Contact him at matthew.klenk@nrl.navy.mil.

Thomas Hinrichs is a research associate professor in electrical engineering and computer science at Northwestern University. His research interests include analogy, machine learning, and design. Hinrichs has a PhD in computer science from Georgia Tech. He's a member of the AAAI. Contact him at t-hinrichs@northwestern.edu.

- General Intelligence Conf.* (AGI 08), IOS Press, 2008, pp. 224–234.
4. N. Cassimatis, "A Cognitive Substrate for Human-Level Intelligence," *AI Magazine*, vol. 27, no. 2, 2006, pp. 45–56.
 5. P. Langley and D. Choi, "A Unified Cognitive Architecture for Physical Agents," *Proc. 21st Nat'l AAAI Conf. Artificial Intelligence* (AAAI 06), AAAI Press, 2006, pp. 1469–1474.
 6. K. Forbus, "Exploring Analogy in the Large," *Analogy: Perspectives from Cognitive Science*, D. Gentner, K. Holyoak, and B. Kokinov, eds., MIT Press, 2001, pp. 23–58.
 7. D. Gentner, "Why We're So Smart," *Language in Mind*, D. Gentner and S. Goldin-Meadow, eds., MIT Press, 2003, pp. 195–237.
 8. K. Forbus, D. Gentner, and K. Law, "MAC/FAC: A Model of Similarity-Based Retrieval," *Cognitive Science*, vol. 19, no. 2, 1994, pp. 141–205.
 9. S. Kuehne et al., "SEQL: Category Learning as Progressive Abstraction Using Structure Mapping," *Proc. 22nd Ann. Conf. Cognitive Science Soc.* (Cogsci 00), Lawrence Erlbaum Associates, 2000, pp. 770–775.
 10. D. Salvucci and J. Anderson, "Integrating Analogical Mapping and General Problem Solving: The Path-Mapping Theory," *Cognitive Science*, vol. 25, no. 1, 2001, pp. 67–110.
 11. K. Forbus and D. Gentner, "Qualitative Mental Models: Simulations or Memories?" *Proc. 11th Int'l Workshop Qualitative Reasoning* (QR 97), Instituto di Analisi Numerica, 1997, pp. 97–104.
 12. H. Simon, *Reason in Human Affairs*, Stanford Univ. Press, 1983.
 13. M. Klenk and K. Forbus, "Cognitive Modeling of Analogy Events in Physics Problem Solving from Examples," *Proc. 29th Ann. Conf. Cognitive Science Soc.* (Cogsci 07), Lawrence Erlbaum Associates, 2007, pp. 1163–1168.
 14. M. Klenk et al., "Solving Everyday Physical Reasoning Problems by Analogy Using Sketches," *Proc. 20th Nat'l Conf. Artificial Intelligence* (AAAI 05), AAAI Press, 2005, pp. 209–215.
 15. M. Klenk and K. Forbus, "Measuring the Level of Transfer Learning by an AP Physics Problem-Solver," *Proc. 22nd AAAI Conf. Artificial Intelligence* (AAAI 07), AAAI Press, 2007, pp. 446–451.
 16. T. Hinrichs and K. Forbus, "Analogical Learning in a Turn-Based Strategy Game," *Proc. 20th Int'l Joint Conf. Artificial Intelligence* (IJCAI 07), IJCAI, 2007, pp. 853–858.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.

Engineering and Applying the Internet



IEEE Internet Computing reports emerging tools, technologies, and applications implemented through the Internet to support a worldwide computing environment.

For submission information and author guidelines, please visit www.computer.org/internet/author.htm