AFRL-AFOSR-VA-TR-2021-0109

Towards Software Apprentices that Learn in Dynamic Domains

Forbus, Kenneth
NORTHWESTERN UNIVERSITY
633 CLARK
EVANSTON, IL, 60208
USA

08/17/2021
Final Technical Report

Air Force Research Laboratory
Air Force Office of Scientific Research
Arlington, Virginia 22203
Air Force Materiel Command

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
17-08-2021	Final	15 May 2016 - 14 May 2021

4. TITLE AND SUBTITLE
Towards Software Apprentices that Learn in Dynamic Domains

5a. CONTRACT NUMBER

5b. GRANT NUMBER
FA9550-16-1-0138

5c. PROGRAM ELEMENT NUMBER
61102F

6. AUTHOR(S)
Kenneth Forbus

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NORTHWESTERN UNIVERSITY
633 CLARK
EVANSTON, IL 60208
USA

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AF Office of Scientific Research
875 N. Randolph St. Room 3112
Arlington, VA 22203

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/AFOSR RTA2

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-AFOSR-VA-TR-2021-0109

12. DISTRIBUTION/AVAILABILITY STATEMENT
A Distribution Unlimited: PB Public Release

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This project made progress on how to create software apprentices, intelligent systems that can operate as collaborators. We did this by developing an account of the representations needed to support flexible learning about dynamic systems, investigating the language understanding and sketch understanding capabilities needed to enable intelligent systems to interact with people as apprentices do, and exploring how to build more autonomous systems, capable of managing their own learning capabilities. This final report describes our progress. We summarize our work on autonomy (including learning via experimentation), natural language, qualitative decision-making, learning from episodic memories, and sketch recognition.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			HAL GREENWALD
U	U	U	UU	50	**19b. TELEPHONE NUMBER** *(Include area code)*

Towards Software Apprentices that Learn in Dynamic Domains

Award #: FA9550-16-1-0138

Final Report

Kenneth D. Forbus and Thomas Hinrichs

Northwestern University

Contents

Abstract

This project made progress on how to create *software apprentices*, intelligent systems that can operate as collaborators. We did this by developing an account of the representations needed to support flexible learning about dynamic systems, investigating the language understanding and sketch understanding capabilities needed to enable intelligent systems to interact with people as apprentices do, and exploring how to build more autonomous systems, capable of managing their own learning capabilities. This final report describes our progress. We summarize our work on autonomy (including learning via experimentation), natural language, qualitative decision-making, learning from episodic memories, and sketch recognition.

Introduction

Intelligent systems are needed for a wide range of complex, open-ended problems. Such systems cannot be pre-programmed to handle all of the situations they will face in advance, since the world is constantly changing and adversaries are constantly adapting. Today's machine learning techniques can work well when a learning problem is well-defined in advance, careful engineering of available inputs has been performed, and massive amounts of data are available. But each of these conditions is a limit to system autonomy. New learning problems arise that were not foreseen by system designers, who are not on hand to design exactly the right representations. Rapid adaptation sometimes requires learning from a handful of examples, or even just one example. The ability to learn quickly is especially important for interactive learning, where people can often learn important lessons from one explanation and a few practice trials – requiring massive amounts of training or experience should not be necessary. Moreover, what is learned needs to be communicable, to ensure trust. When a teacher is training an apprentice, they can discuss what conclusions an apprentice has drawn from the lesson, and provide more guidance and clarification. We think intelligent systems should be more like human apprentices, capable of learning on their own when necessary, but greatly facilitated by learning from people. Creating such intelligent systems that are enough like us to be apprentices, but different enough from us to provide complementary strengths, would be a highly disruptive breakthrough.

Based on evidence from psychology and from our own prior computational investigations, our first hypothesis was that *analogical processing is central to human cognition*, and thus should play a central role in intelligent systems. Our models of analogical matching, retrieval, and generalization provide human-like capabilities, and this project provides further evidence for their utility in building intelligent systems. Our second hypothesis is that *qualitative representations are central to human conceptual structures*. Qualitative representations provide symbolic representations for continuous phenomena, thereby proving a bridge between perception and cognition, a key component of natural language semantics, supports reasoning about problem formulation as well as solutions, and promotes learning general, transferrable knowledge. In this project we expanded our work on qualitative representations to include qualitative decision-making, using qualitative models to guide learning by experimentation, qualitative spatiotemporal histories to support strategic reasoning, and a more general approach to using qualitative representations in natural language semantics. We used the open-source

strategy game Freeciv for most of our experiments, because it incorporates analogs of military, economic, and political phenomena, with complex temporal and spatial dynamics.

Our convention in the sections which follow is that citations in-line refer to publications produced by the project, whereas relevant publications not produced by the project are referred to using footnotes.

Background

Here we summarize relevant background, more information can be found in the publications cited.

Analogical Reasoning and Learning

There is ample evidence that analogy permeates human cognition. We use as our theoretical basis for analogy and similarity Gentner's structure-mapping theory[1]. There is evidence that structure-mapping is used in human reasoning and problem solving[2], visual reasoning[3], conceptual change [4], and cognitive development[5]. Our work uses an *analogy stack* that consists of three operations: *matching, retrieval, and generalization*. We describe each in turn.

Matching is handled by the Structure-Mapping Engine[6], SME, which operates over two structured, relational representations to produce one or more *mappings*, each consisting of a set of correspondences (i.e. what goes with what), candidate inferences (i.e. what would projecting unmapped structure suggest?), and a numerical similarity score. SME's computations have been used to model both analogy and similarity, providing a robust model of human similarity judgments.

Retrieval is handled by the MAC/FAC model[7], which uses a two-stage map-reduce operation for scalability. The first stage uses a cheap, non-structural match to provide a handful of potential retrievals to the next stage for evaluation using SME.

Generalization in long-term memory is handled by SAGE (Sequential Analogical Generalization Engine)[8] and in working memory by SageWM[1]. Analogical generalization takes an incremental

---

[1] Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7, 155-170.
[2] e.g. Jee, B., Uttal, D., Gentner, D., Manduca, C., Shipley, T., & Sageman, B. (2013). Finding faults: analogical comparison supports spatial concept learning in geoscience. *Cognitive Processing*, 14(2), 175-187.
[3] Forbus, K. & Lovett, A. (2021). Same/Different in Visual Reasoning. *Current Opinion in Behavioral Sciences*, 37:63-68, https://doi.org/10.1016/j.cobeha.2020.09.008
[4] e.g. Gentner, D., Brem, S., Ferguson, R. W., Markman, A. B., Levidow, B. B., Wolff, P., & Forbus, K. D. (1997). Analogical reasoning and conceptual change: A case study of Johannes Kepler. *The Journal of the Learning Sciences*, 6(1), 3-40.
[5] e.g. Gentner, D., & Namy, L. (1999). Comparison in the development of categories. *Cognitive Development*, 14, 487-513.
[6] Forbus, K. D., Ferguson, R. W., Lovett, A., and Gentner, D. (2016). Extending SME to handle large-scale cognitive modeling. *Cognitive Science*, DOI: 10.1111/cogs.12377, pp 1-50.
[7] Forbus, K., Gentner, D., and Law, K. (1995). MAC/FAC: A model of similarity-based retrieval. *Cognitive Science*, 19, 141-205.
[8] Kandaswamy, S. and Forbus, K. (2012). Modeling Learning of Relational Abstractions via Structural Alignment. *Proceedings of the 34th Annual Conference of the Cognitive Science Society (CogSci)*. Sapporo, Japan.

stream of examples and produces an analogical model consisting of a set of generalizations and outliers. When a new example is added, MAC/FAC is used to retrieve the closest item. If their SME similarity score is over a threshold, the example is assimilated into a generalization. If the item was a generalization, the new example is added to that generalization, and if it was an outlier (i.e. a previously unassimilated example) then a new generalization is formed. Generalizations are probabilistic relational descriptions, where the probability of a statement is estimated by the number of examples that incorporated something that matched the statement. By maintaining multiple generalizations and outliers, SAGE can handle disjunctive concepts.

Our prior research provided evidence that these computational models are both effective at explaining psychological results and can be effective in building intelligent systems. This project used analogical learning heavily, e.g. in a model of surprise, for learning duration estimates, and for learning tactics and strategic decision-making.

Qualitative Representations

Qualitative representations (Forbus, 2019) provide symbolic descriptions of continuous properties that are suitable for symbolic reasoning, including analogical reasoning. For example, numerical values can be described in terms of how they are changing (e.g. increasing, decreasing, or constant) and how they relate to other values (e.g. greater than, less than, or equal to some key value, like another temperature or pressure in determining flow). Space can also be described qualitatively, e.g. the space inside a well versus outside of it.

Our previous AFOSR project developed several key ideas of qualitative representations that we build upon here:

Type-level qualitative representations[2]provide a form of qualitative reasoning that is more appropriate for open, dynamic domains. Most qualitative representations are instance-level, i.e. involve explicit instantiation of logically quantified representations on specific entities to be reasoned about. This approach has been extremely successful in modeling systems in engineering and science, where for example the set of parts or processes in a system is completely known in advance. However, this is a form of propositionalization, and is worst-case exponential. In domains where the number of potential entities of a given type is indeterminant, propositionalization is an expensive way to reason. Type-level representations provide more concise models that are easier to reason with, and also provide a natural fit for advice expressed in natural language.

Qualitative representations as a language for strategic thinking[3]. Some aspects of strategic thinking involve continuous parameters and processes that can be represented via qualitative representations. For example, agents typically have many goals. Some of these goals trade off

---

[1] Kandaswamy, S., Forbus, K., and Gentner, D. (2014). Modeling Learning via Progressive Alignment using Interim Generalizations. *Proceedings of the Cognitive Science Society*.
[2] Hinrichs, T. and Forbus, K. (2012). Toward Higher-Order Qualitative Representations. *Proceedings of the Twenty-sixth International Workshop on Qualitative Reasoning*. Los Angeles CA, July.
[3] Hinrichs, T., and Forbus, K. (2015). Qualitative models for strategic planning. *Proceedings of the 3rd Annual Conference on Advances in Cognitive Systems*

against each other, requiring setting priorities. Strategic thinking requires both making discrete decisions and tuning parameters such as priorities. Qualitative representations provide a useful medium for framing strategic thinking.

This project used qualitative representations for a new account of qualitative decision-making, for representing goals and tradeoffs for strategic thinking, for capturing spatiotemporal aspects of strategic thinking and episodic memories, and for natural language semantics.

Sketch Understanding

Sketching is a natural modality for people to think through ideas, and to communicate ideas to others. When people sketch with each other, they talk, explaining what they are drawing, so that it is clearly understood. CogSketch[1] provides an interface that enables software to do human-like sketch understanding, i.e. given segmented, conceptually labeled digital ink, it can perform visual processing that enables it to model aspects of high-level human visual problem solving, including spatial analogies.

This project used CogSketch in learning to recognize objects in sketches and for spatiotemporal representations to support strategic thinking and episodic memory.

Companion Cognitive Architecture

The Companion cognitive architecture (Forbus & Hinrichs, 2017) is our platform for experimenting with these hypotheses. Our architectural hypothesis is that intelligent systems are best construed as *software social organisms*[2], which can interact with people using natural modalities, incrementally learning with human-like data-efficiency and inspectability (as provided by analogical processing), both by interacting with people and by conducting their own experiments.

The Companion architecture provides multiple capabilities needed for intelligent systems. Our analogy stack, as described above, is fully integrated. The architecture includes a natural language understanding system[3] which uses a broad lexicon and knowledge resources to produce high-precision representations of statements. This language system uses *narrative functions* to construct task-specific higher-level representations based on domain-general language semantics[4]. CogSketch is also integrated into the Companion cognitive architecture, to provide visual and spatial reasoning services.

The Companion architecture was the platform for all our experiments in this project.

---

[1] Forbus, K., Usher, J., Lovett, A., Lockwood, K., and Wetzel, J. (2011). CogSketch: Sketch understanding for Cognitive Science Research and for Education. *Topics in Cognitive Science*, 3(4), pp 648-666.

[2] Forbus, K. (2016). Software Social Organisms: Implications for Measuring AI Progress. *AI Magazine*, 37(1):85-90

[3] Tomai, E. and Forbus, K. (2009). EA NLU: Practical Language Understanding for Cognitive Modeling. *Proceedings of the 22nd International Florida Artificial Intelligence Research Society Conference*. Sanibel Island, Florida.

[4] McFate, C.J., Forbus, K. and Hinrichs, T. (2014). Using Narrative Function to Extract Qualitative Information from Natural Language Texts. *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, Québec City, Québec, Canada.

Research on Autonomy

An important component of apprenticeship is being able to work and learn with an increasing degree of autonomy, where the system itself takes on more of the work in setting and achieving its learning goals. To understand how to do this, we explored how to learn decompositions of complex domains and how to perform self-directed experimentation to learn the dynamics of the environments that they operate in. This section describes these investigations.

Learning Decompositions of Complex Domains

Decomposing problems to simpler problems is a critical part of problem solving, yet classical planners typically consider only one kind of decomposition: sequencing of primitive actions. In order to reason strategically about high-level properties of a situation and vague, distant future states, systems need to reason about other ways of decomposing problems.

For example, in an extended conflict such as a game, effective strategies may shift over time. By decomposing the game into coarse-grained temporal phases and looking for differences in how competing goals are traded off, it may be possible to induce a strategy from extended training examples. Alternatively, instead of temporal phases, a strategy may be scoped spatially, e.g., by distinguishing a frontier from the interior of a region. Another type of decomposition would be by type of resources (e.g. military vs civilian). A benefit of being able to divide the world up in these ways is that it may make it easier to discern strategic patterns of resource allocation and intent, based on fewer and shorter training examples. Such patterns are harder to find when focusing on individual concrete actions.

In fact, if there are only a few coherent abstract strategies, then learning strategies in a domain may become mostly a matter of discovering decompositions of goals and situations. For example, by comparing offensive and defensive actions, it becomes apparent that defensive actions often occur far in advance of offensive actions. This may be common sense to us, but it is very difficult to tease that out from first principles.

To explore these ideas, we performed three studies to see if abstract patterns actually show up in data from training scenarios and from background domain knowledge. We captured execution traces and snapshots of quantity values from manual gameplay over ten training scenarios and analyzed the relationships between quantities and goals in the first study, between actions and goals in the second, and between background domain knowledge and goals in the third. These were intended to determine whether it is feasible to learn decompositions automatically from examples.

The first study used the Freeciv goal network and its underlying learned qualitative model[1] to guide credit assignment. By treating the game as an optimization problem (a race to achieve technological and military superiority), differences in performance between scenarios can be partly attributed to differences in the relative priorities of different goals at different times. If an expert achieves some milestone sooner than an autonomous learning system, then one way to understand why is to compare the instantaneous quantity values associated with the relevant

[1] Hinrichs, T. and Forbus, K. (2012). Learning Qualitative Models by Demonstration. *AAAI 2012*, 207-213, July.

goals. Credit assignment in this context means regressing backwards through the qualitative influences to subgoals and recursively comparing their values until reaching leaf goals that can be directly achieved or manipulated.

For example, to learn how to rapidly achieve the technology of the Republic form of government in Freeciv, a learner would compare its behavior to an expert's and compare quantitative differences in the subgoals. The expert's trace will likely have a higher science research rate. The goal to maximize science rate has multiple subgoals that trade-off, such as to maximize the number of revenue-producing cities and their individual contributions to the science rate. Regressing down through these influences can reveal factors the learner may have neglected, under-emphasized, or failed to adjust over time as the situation changed. Figure 1 shows a snapshot of a portion of a subgoal network for a training scenario at a particular turn.

What this study showed is that the basic idea of using the qualitative model to guide credit assignment makes sense, but that there are difficulties that must be overcome before it can be fully automated. In particular, qualitative influences in a model are not limited to simple additive factors. Some are multiplicative, like the influence of number of cities on research rate or the fractional percentage of trade revenue to be converted to research. Consequently, it is not possible to make a simple Pareto chart, for example, to assign credit. We believe there are sufficiently few influence patterns that this will not be a major concern.

The second pilot study sought to determine whether decompositions can be discerned in expert traces based on explicit actions taken. Here, the input is not quantitative relations, but categories of actions and objects acted on over time. Roughly, the question is can we learn to do factored planning by inducing the factors from example?

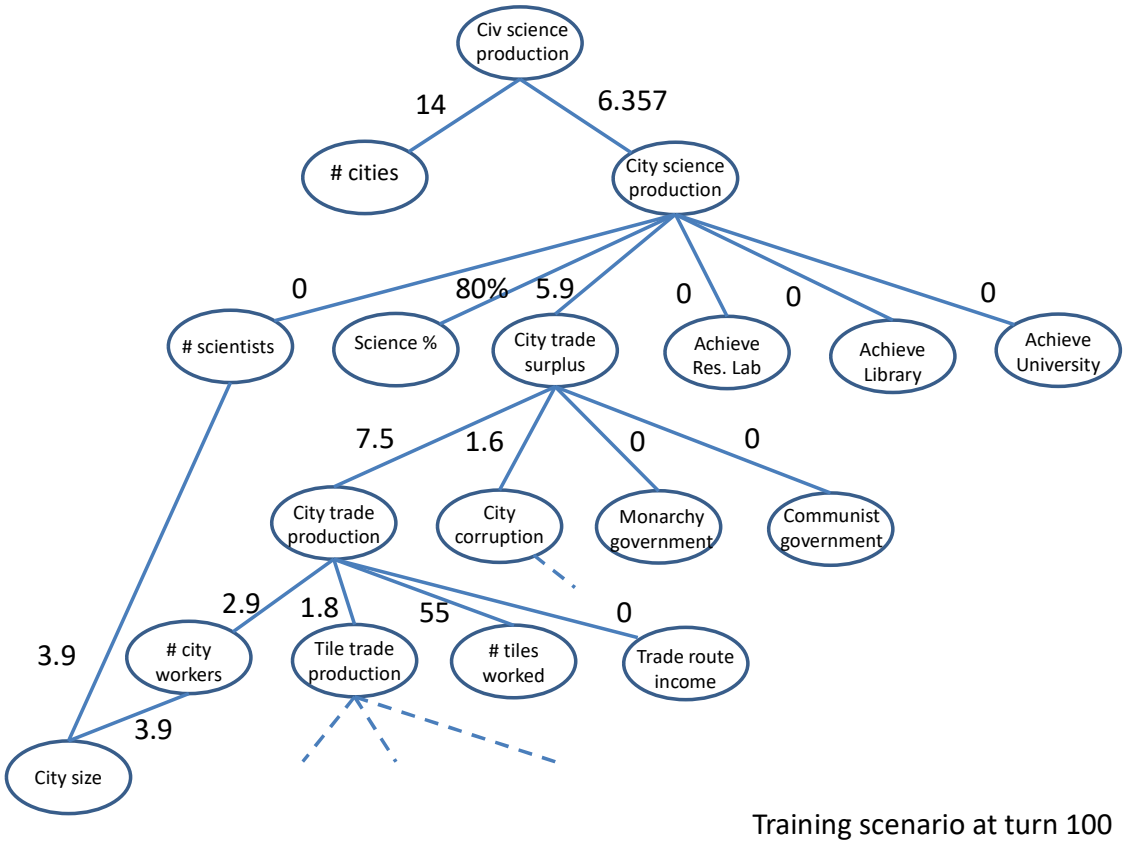


Figure 1: Using the qualitative model to guide credit assignment

As before, the raw data is the execution traces of the same ten training scenarios. The analysis took a known goal tradeoff and gathered all the actions that pursue the different competing subgoals and indexed them by game turn. Building this index required first associating the recorded actions with the low-level goals they are likely to pursue. This was done simply by matching the effects in the action models to achievement goals in the goal network. Following the subgoal links allowed the actions to be associated with the higher-level tradeoffs.

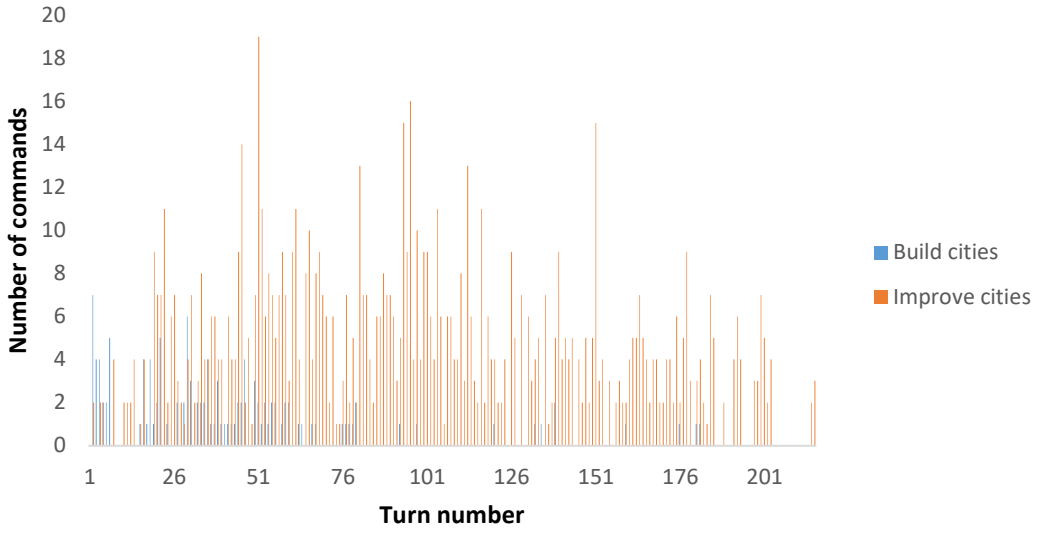


Figure 2: Build actions vs Improvement actions across all training scenarios

We first considered whether a known strategy, build-grow, is actually detectable in the data. Figure 2 shows the distribution of build city commands vs the actions they trade off with that construct city improvements. Because this data is aggregated over ten scenarios, the pattern is

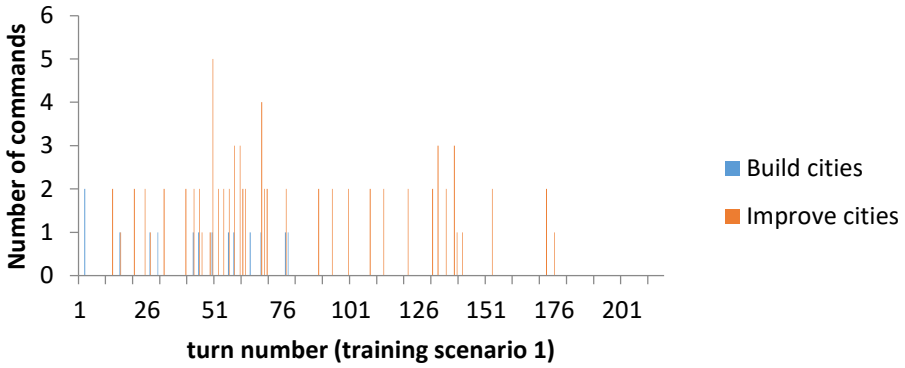


Figure 3: Build actions vs Improvement actions for training scenario 1.

not quite as clear as it is for an individual scenario (see Figure 3).

Figure 4 shows a decomposition by offensive vs defensive actions. These are simply actions pursuing subgoals of preserving existence and conquering opponents. Here, it is readily apparent that defensive actions are often taken long before any actual combat.

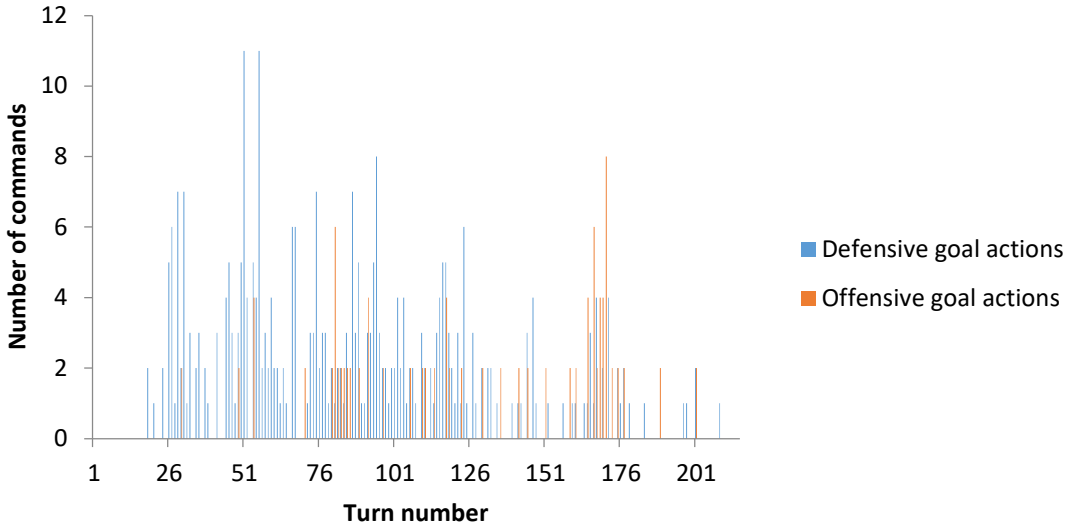


Figure 4: Defensive actions anticipate combat

This second study suggested that some strategic considerations may be detected by searching through potential decompositions, but only if the relevant relations are stored in the training cases to begin with. For example, it was not possible to explore spatial decompositions, because the relevant spatial relations were not stored (e.g., distance to the capitol, distance to the frontier).

The third study was an attempt to determine if background knowledge about generic systems in the world can help to decompose the distant goal ("winning") into more short-term subgoals without requiring concrete, detailed plans to the end state. For example, if a system can recognize entities in its environment that serve the function of path systems, then it can assume that constructing these entities will help in pursuing the goal of enhancing mobility, a kind of efficiency, which is inherently desirable. Alternatively, something in the domain that corresponds to "money" is thereby a fungible resource that can be accumulated and is desirable.

If such correspondences can be inferred, then this provides a way to connect some quantities and actions to goals, even if those goals cannot be directly associated with a top-level goal of winning. This is important because in real life we seldom have (or are aware of) a single top-level goal. We adopt medium-term goals as surrogates for end-states. By identifying the potential existence of systems in a domain, an agent can pursue productive tasks without having a complete goal model.

An important question is how much domain-specific knowledge must be encoded to enable the recognition and pursuit of intermediate, system-level goals? Can a system identify resources in the domain and actions that convert resources from one form to another? Understanding what constitutes a resource is critical for recognizing or operationalizing general resource conservation and exploitation strategies.

In Freeciv, the domain is intended to suggest real-world correlates. Roads and Railroads are idealizations of actual path systems. GoldPoints correspond to money in some game-specific denomination. Other quantities can be accumulated and depleted, suggesting they be treated as resources. Some events can be interpreted as signaling an improvement of an entity (e.g., upgrading a unit or achieving veteran status).

What we found was that a sparse connection between domain entity types in Freeciv and their real-world counterparts is usually sufficient to infer the existence of a system. For example, it is sufficient to indicate that a Freeciv "road" is a kind of Roadway in order to determine that a transportation system is something that can exist in the domain and that building paths between destinations is generally desirable. Background knowledge about path systems already exists in our NextKB knowledge base and did not need to be constructed from scratch.

Likewise, determining what serves the role of money requires only a single generalization link, and from there it is possible to establish the desirability of accumulating it, spending it when necessary, and building infrastructure (e.g., marketplaces, banks) that enhance the accumulation of money. These are all inferences that human players make automatically. If it weren't for the English labels and familiar icons, humans would be reduced to blind guessing much like knowledge-free programs.

We identified several inferable features that support recognizing systems, including:

- Constructible entities require durative actions to create
- Growable entities grow on their own
- Consumables can be destroyed
- Monetary fluents are fungible resources
- Economic events enhance or facilitate resource conversions
- Transportation events move entities under their own power or as cargo

These abstract features and dimensions justify more in-depth searches for systems that can be constructed and goals surrounding those systems. Identifying what constitutes a resource in a domain is an important step in operationalizing general strategies for investing, conserving, and converting resources efficiently, independent of particular domain-level goals.

This third study suggested that abstract background knowledge can be operationalized, or re-cast in a domain-specific way without incurring a large burden of hand-crafting domain knowledge. In the existing Freeciv agent, very few inference rules are defined at the domain level. The vast majority are either domain-independent or learned from demonstration. Analyzing the domain to search for potential systems is another way to leverage or transfer available background knowledge to extend the goal network.

The focus on decomposition is intended to define a search space over which a learning agent can experiment to optimize behavior. If the sole evaluation function is winning the game, experimentation would not be practical unless the game were radically foreshortened and simplified. Inferring decompositions from small training scenarios and intermediate system goals from analysis of the domain and background knowledge should make active learning and experimentation more tractable.

Experiments with self-directed experimentation

A major focus of this research is self-directed learning through experimentation and reinforcement. We see this as a logical endpoint of apprenticeship in which the learner takes responsibility for deciding what to learn, what to vary and what to observe, and designs experiments to control that process. Here, experimental design is a way to reduce the challenges of credit assignment. Our eventual goal is for a Companion to operationalize abstract strategies in a game domain using a qualitative model acquired through observation and advice. In other words, starting with a qualitative model, systems should be able to learn appropriate quantitative tradeoffs to behave more strategically.

Our approach extends our earlier work in representing strategies through qualitative process models. That work showed that the same process model formalism could influence levels of goal activation to drive behavior. Building on that, we hypothesize that strategies can be defined in terms of four orthogonal factors: *Prioritizing, Scheduling, Decomposing,* and *Qualifying,* which we refer to as the **PSDQ** operators. A strategy can prioritize one goal over another by influencing their activations. It may schedule such decisions in ways that achieve just-in-time behavior (hence investigation of learning durations of actions below). It may decompose a type-level goal by properties of its entities in order to, e.g., specialize different resources based on intended use. It may qualify influences or decompositions based on qualitative states of the situation. For the purposes of our research, we think of these as separable learning operators (the **PSDQ** operators), through which an agent can search for effective refinements of general strategies.

One important task is to learn to qualify strategies through active experimentation. Given a qualitative model with tradeoffs, that means picking different ratios for tradeoffs to gradually learn more optimal gameplay. Of course, in any real game, goal tradeoff ratios are not actually an independent variable that can be set directly. If the tradeoff is between high-level abstract goals, such as offense vs. defense, then a large part of the problem is figuring out how to achieve the desired balance given the lower level actions and quantities available. This problem is compounded by the issue of maintaining control over exogenous factors, such as adversaries that interfere with carefully planned experiments, stochastic outcomes, and incomplete knowledge. As it turns out, these are all problems that make Freeciv a challenging domain for exploring experimental design. In order to focus more directly on autonomous experimental learning, we elected to start with a somewhat simpler game in a different domain. This had the additional benefit of demonstrating the generality of the approach and helps to avoid over-fitting to any particular game.

Rather than spend a great deal of time designing a new game from scratch or engineering an interface to a commercial game, we re-purposed the quantitative model behind a training simulation developed at the Institute for the Learning Sciences in the early 1990's. The simulation was called HRM, for Human Resources Management. It was intended to teach the principles of human resources by having the student run a simulated company for some period of time and make decisions about hiring and firing, training, promoting and giving raises such that

she balanced the need for high productivity and profits with employee morale. Driving employees too hard or failing to develop their skills could lead them to quit, sending profits into the red. Students learned to navigate these tradeoffs and keep the company afloat by knowing when to take action, to whom, and when to leave well enough alone.

In re-implementing HRM for Companions, we encoded the quantitative equations as backchaining rules in a slightly modified interpreter for General Game Playing[1]. The objective of the game is to avoid bankruptcy over a simulated span of twenty months. The primitive actions available to the player are to hire an applicant into a position, to fire an employee, assign an employee to one of several training courses that teach different skills, to promote, give a raise, set or remove probation, and perform an evaluation. The goal tradeoffs in this domain are between 1) minimizing salaries to keep down labor costs and maximizing salaries to improve attitude, 2) minimizing training to keep down costs and maximizing training to improve competence, and 3) minimizing the staff (number of employees) and maximizing the staff to maximize production. Ultimately, a good strategy is to maintain a light touch, but address problems early, before they become serious. Not surprisingly, the initial default behavior of the agent is to do everything that might conceivably address an active goal. That strategy does not end well.

The active learning technique that we developed is a combination of experimental design and reinforcement learning. Because the ultimate goal is to improve performance on a task, learned knowledge is fed back into the performance system to hill-climb toward more optimal behavior. However, rather than learning an action policy that maps directly from states to effective actions, we learn *constraints* on behavior that vary in the specificity of the state description and in the generality of the actions to take or avoid. In fact, the states are qualitative state descriptions, and the actions may be concrete primitives with ground arguments, or abstracted actions or decisions with variablized arguments. This flexible mapping representation is used both as an action policy and as a way to specify experimental controls.

Experimentation comes into play in two ways: 1) refining an action policy in response to failures and sub-optimalities, and 2) exploring the space of tradeoffs in the qualitative model. First, in response to failures or suboptimal play, credit assignment is invoked to identify concrete actions and events that led to failure and action learning goals are posted. Two control conditions are produced that correspond to the most specific state and action, and to the most general. In this case, it tries to refine the action policy by creating a control condition between the two extremes in a manner similar to version spaces and runs the experiment to discover the appropriate constraint. For example, with no learned knowledge or controls, the agent lost fairly quickly by trying almost all actions, including firing everybody (see Figure 5). The most specific constraint produced was “don’t fire SylviaNasar when the capital is less than $10,000” and the most general is “never fire anybody”. Experiments, then, generalize the specific constraint by either abstracting the action to omit the particular employee, or drop a condition from the state description. This refinement may be iterated until a stable action policy is discovered.

[1] Genesereth, M. & Thielscher, M. (2014) *General Game Playing*, Morgan & Claypool Publishers.

Experiments to explore tradeoffs are driven by systematically varying goal activations in a pairwise fashion. This was done in a simple, low resolution way by forcing the tradeoff all the way in favor of one goal over the other, and then trying the tradeoff in the opposite direction. We used the same qualitative control condition representation, except that the action is an action to set a goal tradeoff allocation. In HRM, this meant scheduling an experiment to keep salaries low, followed by an experiment that maximizes salaries, then deciding which condition performed better and adopting that constraint before moving on to the next tradeoff to investigate.

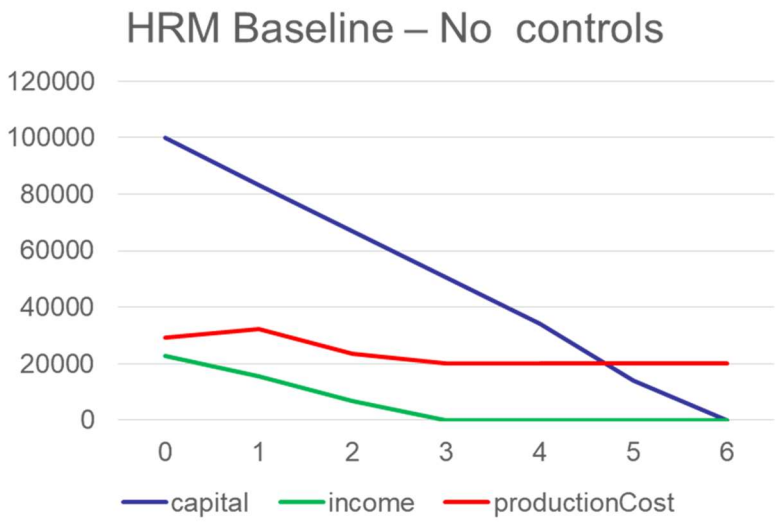


Figure 5: Baseline performance on HRM

Figure 6 shows how the system performed. Starting with no controls, it quickly learned in the first trial which actions fail to improve their goal quantity and constrains their use. The first six trials explored the goal tradeoffs, preferring first one, then the other goal and cumulatively adopting the better performing tradeoff. Experiments five and six explored the tradeoff in staff size, first firing everybody, then hiring two people. Trials seven to ten continue to hire two additional employees, varying the order in which they're hired and the positions to which they are assigned.

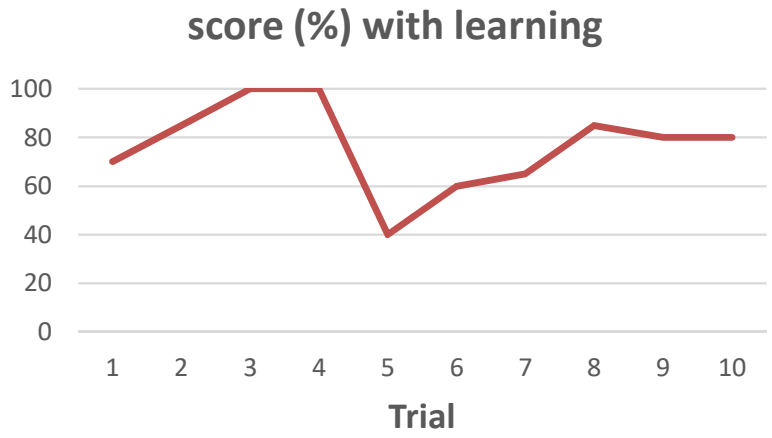


Figure 6: Performance on HRM with incrementally learned controls

These experiments provided some important insights for learning how to close the feedback loop and systematically explore tradeoffs and refinements of action policy constraints. The learning curve is highly non-monotonic, as would be expected for a system that purposely explores extreme tradeoffs that may turn out to be very bad, such as firing everyone. On the other hand, it improves its performance in a very small number of trials, unlike standard reinforcement learning.

We continued our investigation of autonomous experimentation and reinforcement learning in HRM, the Human Resources Management game. The primary issue we explored was how to support autonomous experimentation using a qualitative model to learn in a small, realistic number of trials. We found that a qualitative model provides three key benefits:

1) It enables better credit assignment by specifying causal influences that allow the learner to reason about quantities over time
2) It supports better exploration than random state-space search by allowing a learner to set up experimental controls that constrain exploration and reduce ambiguity in credit assignment, and
3) It allows adaptive state definitions such that learned action policies are specified by relations between quantity fluents, rather than by arbitrarily carving continuous quantities into large numbers of discrete states.

Starting with a qualitative model and a goal network generated from that model, the system learns conditions under which it is or is not advisable to take particular actions in the game. For example, one way to improve an employee's attitude is to give them a raise, but that is not appropriate if the employee is already overpaid, or if the company cannot afford it. Moreover, the criteria for giving a particular employee a raise are not contingent on the competence or attitude of some other employee. The qualitative model enables the learner to determine which quantities are or are not salient for a given action and to devise parametric experiments to refine the ranges of quantity values that rule the action in or out.

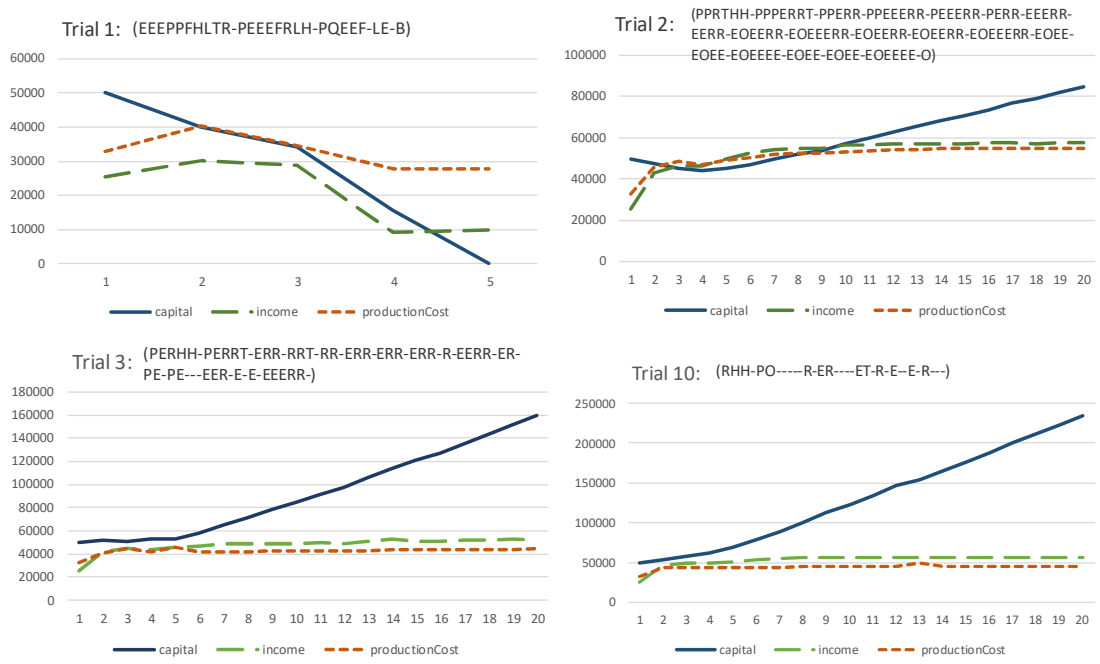


Figure 7: Action learning trials 1-3 & 10. The action abbreviation key is: **E**valuate, **P**romote, **H**ire, **F**ire, **R**aise, **T**rain, **L**awsuit, **O**verpaying, **B**ankruptcy

Figure 7 shows the performance of the system when learning the conditions for actions as described above. Starting with $50,000, the agent's objective is to run the company for twenty simulated months, avoid bankruptcy, and maximize profit. It started out with only the qualitative model and goal network to guide it, which meant that it knows what quantities an action influences and how those quantities relate to goals in isolation, but not how to balance competing goals to optimize overall performance. Consequently, in Trial 1, the agent fired too many employees and the company went bankrupt in month five.

At this point, post-mortem credit assignment looked at the proximal cause of the loss (capital reaching zero) and looked back to the last action that indirectly influenced the rate of change of capital (firing someone). It introduced an action policy for that action and describes the state of that policy rule in terms of the values of the salient quantities with respect to capital. Subsequent trials then made use of and refined this action policy that specified when it is a bad idea to fire someone.

Trials 2, 3, and 10 show that the agent continued to learn and improve its profitability while simultaneously reducing the number of actions taken. (Trials 4-9 are omitted for brevity, but show monotonic improvement)

In addition to these action learning experiments, we also investigated learning to quantify goal tradeoffs more directly. Here, the experimental controls were different in that they determine the relative goal activations of the competing goals. Recall that this domain has three goal tradeoffs: 1) maximize salaries to improve productivity vs. minimize salaries to reduce production costs, 2)

maximize employee training to increase competence vs. minimize training to reduce costs, and 3) maximize the number of employees to increase production vs. minimize the number of employees to reduce labor costs. By experimentally adjusting the relative activations of these goals, the learner controlled the relative frequency of actions to pursue those goals and was able to explore these tradeoffs independently.

Figure 8 shows the results of tradeoff learning trials. In the first five trials, it lost via bankruptcy, although it learned action policies for unsuccessful actions. In Trial 6, it emphasized maximizing the number of employees and won by hiring additional employees early in the game.

These tradeoff learning experiments demonstrated one way for a learner to experiment at a more abstract level than that of primitive actions. These results were presented at the Qualitative Reasoning Workshop and the Advances in Cognitive Systems conference (Hinrichs & Forbus, 2019a,b).

Figure 8: Tradeoff learning trials

Representing Tactics

The next step in exploring experimentation involved returning to the Freeciv domain and focusing on higher-level domain-independent tactics with which an agent can experiment and learn. The work in HRM focused on experimenting with when to apply different primitive actions and how to resolve quantitative tradeoffs at an abstract level, it rapidly became clear that in order to improve performance in Freeciv, an agent would have to experiment with intermediate-level aggregate behaviors, or tactics.

Rather than encode Freeciv-specific tactics, we have been developing domain-independent military and economic tactics and devising ways of interpreting this generalized, event-centered knowledge in terms of Freeciv-specific primitive actions. We see this as a critical step, not only to self-directed experimentation but also as a step toward learning from real-world historical precedents and learning through language-based interaction and advice.

The tactic representation is more than simply a sequence of primitives or an HTN plan. As with our representations for strategies and for experimental conditions, tactics may be quantity conditioned, so that they can be active or inactive, depending on the situation. Unlike an HTN plan, the participant roles are explicitly represented, making it easier to specify coordinated activities. Because these tactics are hierarchical and have explicitly represented subevents, it is possible to specify more sophisticated and precise temporal constraints than the simple linear sequencing of plan steps. Perhaps most importantly, because they are represented in a neo-Davidsonian manner (i.e., frames with role fillers), it becomes possible to incrementally refine a tactic in an explicit representation of intent (a Course Of Action microtheory, or COA). In other words, as resources become available and possible targets become apparent, the COA can be progressively filled in until a tactic is operational. At that point, the tactic can be scheduled on the Companion agent's agenda like any other task.

Representing tactics: We have been focusing initially on two types of tactics to prove out these ideas: search-and-destroy and building path networks. We chose these to start with because they address areas in which our current agent is especially weak: offensive actions and investments in distributed infrastructure. That is because it is extremely difficult to synthesize such behavior from a qualitative model and a set of action primitives. By starting with larger building blocks, it becomes a much more plausible learning task. The challenge has been to make those building blocks be independent of the particular game domain.

For example, here is a portion of the representation for the tactic of search and destroy:

```
(isa SearchAndDestroy TacticType)
(comment SearchAndDestroy "A SearchAndDestroy mission is the extended event of looking
for targets, maneuvering, closing on, and destroying them.")
;;; Connect to rest of ontology
(genls SearchAndDestroy MilitaryTactic)
(genls SearchAndDestroy OffensiveTactic)
(isa SearchAndDestroy DurativeEventType)
(genls SearchAndDestroy PurposefulAction)
;;; Participants in this tactic
(participantType SearchAndDestroy performedBy MilitaryAgent)
(participantType SearchAndDestroy eventOccursAt GeographicalRegion)
(associatedRoleList SearchAndDestroy (TheList performedBy eventOccursAt))
;;; Types of subevents – there can be lots of instances of these
```

```
(properSubEventTypes SearchAndDestroy TargetPicking)
(candidateProperSubSituationTypes SearchAndDestroy
                MilitaryManeuver-Offensive)   ; getting into position
;;; Sometimes one waits for best advantage, surprise, etc.
(candidateProperSubSituationTypes SearchAndDestroy Waiting)
(properSubEventTypes SearchAndDestroy ClosingWithEnemy)
(properSubEventTypes SearchAndDestroy DestroyingAnEnemyForce)
(focalProperSubsituationTypes SearchAndDestroy DestroyingAnEnemyForce)
;;; Repeat for more targets
(properSubEventTypes SearchAndDestroy RescheduleParentTask)
;;; Start of temporal constraints among these tasks
(startsAfterEndOfInSituationType SearchAndDestroy
                                 MilitaryManeuver-Offensive TargetPicking)
(startsAfterEndOfInSituationType
                                 SearchAndDestroy Waiting MilitaryManeuver-Offensive)
[…]
;;; Find new enemy to destroy after the current one is finished off
(startsAfterEndOfInSituationType SearchAndDestroy RescheduleParentTask
                                 DestroyingAnEnemyForce)
[…]
```

This summary leaves out the relationships which link participants in the tactic to their roles in the instances of the types of events that occur within the tactic, as well as the specifications of the constituent events. Importantly, the temporal constraints shown above are used by the Companion agenda mechanism, so that once a search and destroy begins, it will be executed by that mechanism.

More flexible control: One implication of combining these tactics with our existing Freeciv planning mechanism is that we have been forced to re-think the entire control structure of the game player. Instead of a rigid turn-taking loop that is specific to Freeciv, we moved to using the agenda built into the Companion architecture. The benefits of this are that it affords greater flexibility, since tasks with extended duration can overlap, their order of execution can be dependent on explicit priorities, and multiple, alternative courses of action can be reasoned about and compared. The main drawback is that this flexibility comes at the price of complexity. The control system is now more like an operating system than a simple plan execution system. We have devised techniques for invoking the next turn when the agenda becomes quiescent, re-scheduling recurrent tasks, and deferring tasks across multiple turns. These techniques have been propagated to the Freeciv domain and support experiments in tactical decision making.

Learning Tactical Decision Making

In the final year of this project, we explored the autonomous learning of tactical decisions. The main idea is that an effective way to learn complicated behavior is to decompose that behavior into constituent decisions and learn them independently. The tactical representations of the previous year make decomposition straightforward, but the complexity of the situations make the simple experimentation techniques of HRM less feasible. We next describe how we addressed this problem and the resulting performance of the learner.

Our initial approach was a direct extrapolation from the previous HRM learner. In this case, we converted decision cases into quantitative relationships between fluents and constructed policies to rule in or out different decisions. We found that this performed poorly for several reasons:

1. It over-generalized and quickly ruled out any attacks at all based on early failures.

2. It did not construct relationships between features, thereby ruling out what we believed would be an optimal policy (attack strength of attacker > defend strength of defender)
3. Even when we hard-coded a policy that we expected to be optimal, it still performed poorly (i.e., worse than baseline)
4. It did not learn from observing opponents
5. It was unable to resolve conflicting or inconsistent policies

The last is a problem because combat in this game is fundamentally stochastic. Even with a perfect policy, it can still fail. That failure then led it to rule out attacking anyone and never try again. This quickly led to behavior that was purely passive.

Some of these problems might be resolvable through the experimental design mechanisms developed previously, but in this case, there were many more quantities and features to manipulate and would have required far more trials than was feasible.

We made several modifications to change the way generalization and action filtering worked:

1) Instead of analytically producing the minimal quantitative case description as a QP quantity condition, we fed the case facts into SAGE to produce analogical generalizations. These generalizations retain the original cases as examples and use them to compute probabilities associated with different features.

2) We now compute quantitative relations between fluents. To avoid a combinatorial explosion of such constructed features, they are limited to relationships between fluent quantities with the same dimension. For example, it will construct a `greaterThan` relation between the attack strength of the attacker and the defense strength of the defender, but it won't construct a relationship between the food upkeep requirements of a unit and its movement rate, or other such unrelated quantity. This produced a manageable set of second-order features that included highly relevant relations.

3) We infer the adversary's actions based on events captured in the game. In particular, we added a piece of domain knowledge (eventActionMapping) that relates the events DefenderDestroyedEvent and DefenderSurvivedEvent to the doAttack command that would have led to the command. The mapping plus the information recorded about the particular event is enough to reconstruct the concrete action that would otherwise be opaque to the game player. Learning from the adversary's successes and failures is an effective way to avoid a major pitfall of incremental learning, which is prematurely suppressing actions that could lead to learning. Explicit experimentation is another approach, but we found it impractical in this domain.

The way the SAGE-based filtering works is that tactical actions in the game are mediated by an explicit decision - a go/nogo decision that filters the action if it does not look promising. It does this by constructing a probe case from sampled quantities consisting of the intrinsic and extrinsic properties of the arguments of the action. It then invokes SAGE to classify the probe with respect to two analogical models: the generalization pool of successful cases of the action and the generalization pool of failed cases. Whichever generalization pool provided the item that more closely matches the case determines whether the action proceeds.

Generalization is driven by credit assignment that is triggered from game events. Credit assignment attempts to trace back from the event to a decision that may have been made much earlier. For example, target selection within a SearchAndDestroy tactic may have set a unit in pursuit of an opponent's unit long before the actual doAttack command was issued. When possible, those early decisions are what we would like the system to learn. In other cases, there are simple targets of opportunity which lead to direct go/nogo decisions that we describe here.

Experimental results: We ran a sequence of ten autonomous games up to turn 100 and measured the number of successful and failed attacks and defenses (See Figure 9). Although defensive combat is not itself a decision, our purpose in including these is to ensure that a decision to refrain from attacking doesn't simply mean that the unit is immediately destroyed in a defensive engagement. The metric that matters is the total number of units lost to the total number of enemy units destroyed.

In the baseline condition, the game was played with no learning at all and simply attacked whenever an opposing unit was in range. In the incremental learning condition, each combat interaction case was added to an analogical generalization of successful or unsuccessful attacks which informed future decisions about whether or not to attack a target unit. Trials 1-10 correspond to ten distinct scenarios, or maps, such that differences in performance across conditions is due to differences in decision making in the game, and not due to differences in terrain or initial conditions.

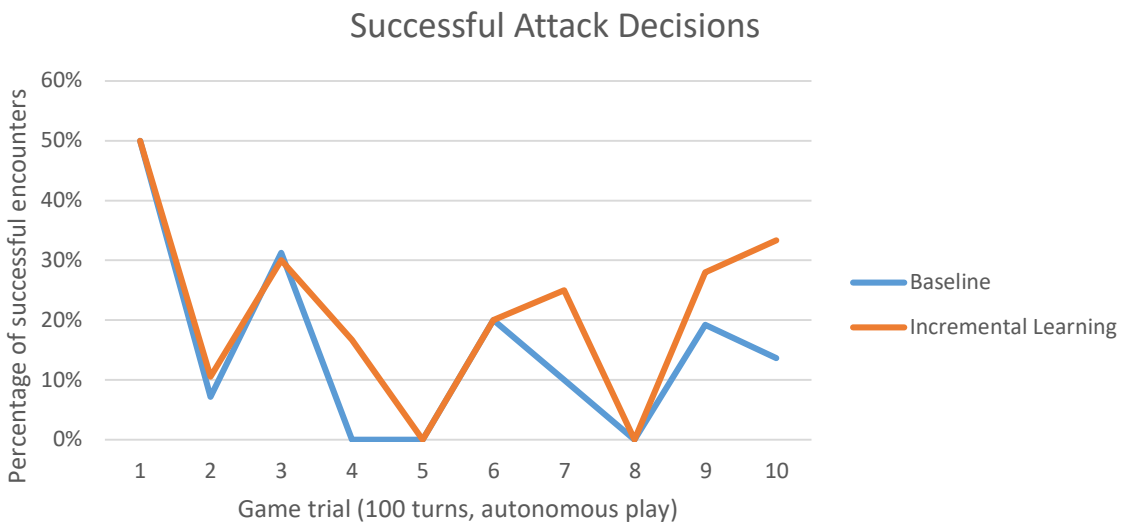


Figure 9 Combat performance

Although the performance in absolute terms is not inspiring, the improvement over the baseline is evident as early as the fourth trial. At the end of the tenth game, the generalizations comprise 112 cases of attacks or defenses. Large variations in performance between trials is mostly due to the proximity of aggressive adversaries on different maps. Improvement over the baseline shows

the effect of making better decisions about when to take the initiative in attacking vs. holding off with a passive defense.

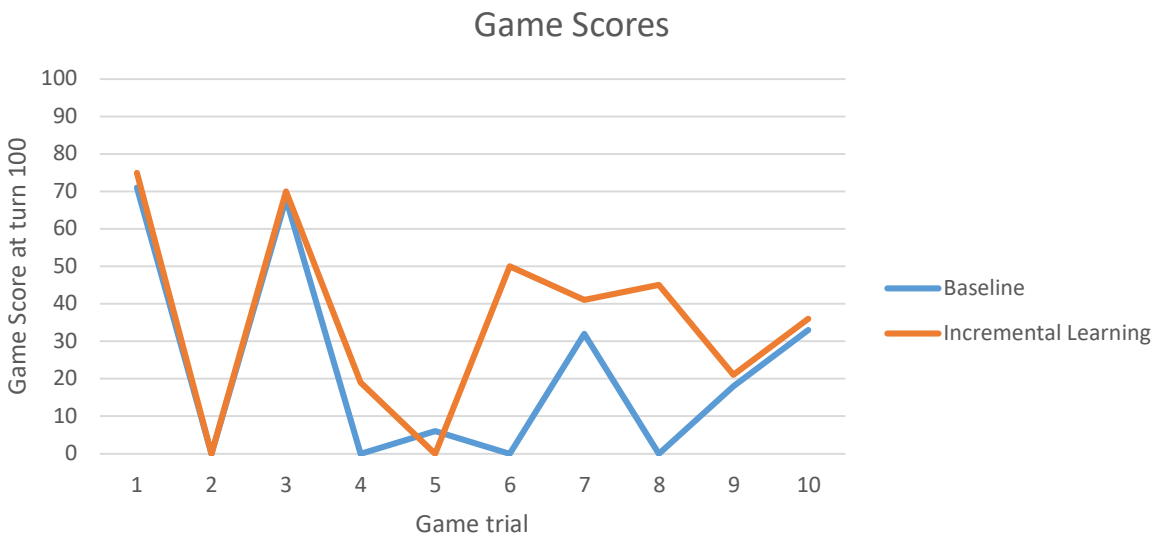


Figure 10: Freeciv game scores at turn 100

Figure 10 shows the overall score provided by the Freeciv server in its post-game report. The score rolls up an assessment of a variety of factors representing the health of a civilization. What this shows is that the improvement in combat outcomes does not come at the expense of population, the economy, or technological research. This is important because one of the assumptions of our approach is that tactical decisions can be learned independently.

These experimental results suggest that tactical decisions can be learned when there is clear feedback in the form of events that can trigger credit assignment. Credit assignment reasons about these game events and their domain-independent parent types in order to determine which goals have been satisfied or violated, if any. It then uses these goals to identify the tactical decisions that were made and whether those decisions were successful or not. This determines whether the decision case should be added to the generalization of successful actions or unsuccessful actions.

While this is a more general approach to credit assignment than was used with HRM, it is still limited by the reliance on explicit events. There are many decisions, such as determining what to build or what government to select, that simply lack that clear feedback. Ultimately, those decisions would require the system to learn when and how to assess its own performance, and to track different high-level metrics. The longer-duration episodic memories discussed below should provide a useful representation for this. Alternatively, this may be viewed as a critical role for apprenticeship learning, where credit assignment is provided by an instructor through explicit advice. This is likely to be more useful advice than simply telling the system what to do next. Learning to assign credit from advice would be a natural next step in future research.

Language use by Software Apprentices

We explored three aspects of natural language in this project, i.e. using abduction in language understanding, how to use analogy for reference resolution in natural language understanding, and an improved frame representation for qualitative representations in natural language semantics.

Representations and Reasoning for Apprenticeship Dialogues

Our prior work on understanding the language capabilities needed by apprentices has focused on off-line learning. This has included advice about effects of actions[1] and reading simplified syntax versions of chapters from the Freeciv manual[2]. A core capability of apprentices is the ability to engage in natural language dialogue with their trainers. We have made progress on this in several ways:

1. We have rebuilt the abductive reasoning capabilities in the FIRE reasoning engine, so that they are cleaner, more robust, and support backtracking. That is, annotations about alternative abductive proofs and nogoods are now recorded in the system's working memory, so that backtracking points are explicitly represented.
2. The design of spatiotemporal representations for episodic memories, outlined elsewhere in this report, should yield tractable and concise representations of long intervals of play, including entire games. This should support communication with trainers as well as the system's own learning.
3. Since apprentices must respond using natural modalities, we implemented a robust natural language generation system, which uses compositional templates to translate internal representations into English text. These templates can draw upon FIRE's reasoning capabilities in producing explanations. For example, one of the tradeoffs found via automatic analysis of the qualitative model and goal network is summarized as: "There is a partial abrupt tradeoff[3] between maximizing the manufacturing capacity of cities the player owns and minimizing the city pollution of cities the player owns."

Future work will involve building out question-answering capabilities, using our prior work on narrative function and our recent work on analogical Q/A training[4]. We see four contexts where dialogue is important for apprentices in constructive dynamic domains:

1. *Discussing domain theories*. Trainers need to know what models their apprentices have, including both causal models and experiential models. An example of the former are

[1] McFate, C.J., Forbus, K. and Hinrichs, T. (2014). Using Narrative Function to Extract Qualitative Information from Natural Language Texts. *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, Québec City, Québec, Canada.
[2] McFate, C., and Forbus, K. (2016). Scaling up Linguistic Processing of Qualitative Process Interpretation. *Proceedings of the Fourth Annual Conference on Advances in Cognitive Systems*. Evanston, IL.
[3] In our representations, a tradeoff is partial if it can be resolved by favoring one goal type for some instances and the other goal type for other instances. A tradeoff is abrupt if it cannot be done incrementally over time. Choosing to build factories in some cities and not others – a set of discrete actions – is how this tradeoff would be handled.
[4] Crouse, M., McFate, C.J., and Forbus, K.D. (2018). Learning from Unannotated QA Pairs to Analogically Disambiguate and Answer Questions. *Proceedings of AAAI 2018*.

domain tradeoffs, as per above, and an example of the latter are the priors for different outcomes for entering a hut as described below.

2. *Discussing abstract scenarios*. Trainers need to be able to pose abstract scenarios and ask broad questions about what makes sense in them (e.g. the Freedonia example below).
3. *Discussing particular situations during gameplay*. This includes asking about the apprentice's intent for a particular unit or rationale behind an action, as well as giving advice about alternatives. Suggestions about where attention should be paid can be used to help the system build better models of what to monitor ("Have you noticed that you are going bankrupt?").
4. *Discussing a game afterwards*. After-action reviews are a standard practice in the military. Being able to compare and contrast broad-brush descriptions of entire games, by abstracting away properties of the episodic memories that describe them, could be used to help evaluate the effectiveness of experiments, e.g. win/loss record for military engagements, decision-making strategies for growth (Forbus & Hinrichs, 2019).

Our Companion dialogue management capabilities are inspired by Allen's collaborative problem solving model[1]. In future work we hope to extend these capabilities to handle these four kinds of dialogues in dynamic constructive domains, using Freeciv as a testbed.

Using Analogy for Reference Resolution in Natural Language Understanding

Software apprentices need to be able to communicate with people using natural language. A long-standing problem in natural language dialogue is *reference resolution*, which is the problem of determining what entity or situation in the world is being referred to by language. One troublesome phenomenon is the *near miss problem*, which are definite descriptions that do not perfectly match their intended referent but are close enough to be understood by the listener. Consider this exchange:

A: "Did you hear about the man who jumped off Swift Hall?"
B: "He didn't jump. He was pushed."

B was able to determine what A was referring to and offer a correction. This indicates that exact matching, while a popular approach to reference resolution, is too inflexible for reference resolution. Our approach is to use analogical matching. That is, the semantic interpretation of a referring expression is treated as a case. Our MAC/FAC model of analogical retrieval is used to retrieve the most similar entities[2]. The entity in the retrieved case that corresponds to the discourse variable in the probe is treated as the result. A threshold is used so that very weak retrievals are ignored, otherwise it would always propose something.

This model has several attractive properties. First, it handles both exact matches and near-misses. When there are near-misses, the alignable differences found by SME (e.g. "jumped" versus "pushed") provide the information which can be used to evaluate whether it is close

[1] Allen, J., Blaylock, N., & Ferguson, G. (2002). A problem solving model for collaborative agents. *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*. Bologna, Italy: ACM Press.

[2] We treat actions and events as entities as well, using a neo-Davidsonian representation.

enough, or to provide a correction, as in the example above. Second, ambiguity occurs when multiple referent cases are retrieved, which provides grist for using further heuristics for disambiguation. To evaluate this model, we used the TUNA corpus[1] which was designed as a benchmark for systems that generate referring expressions. Each item consists of a set of entities and a human-authored description referring to one of them. The descriptions were collected by presenting each participant with a 3x5 grid containing pictures of seven entities and asking them to describe the one marked. The attributes mentioned in their responses were used as annotations for the stimuli, to support computational experiments. The corpus contains two domains:

- People: 360 descriptions where the entities are black and white photographs of people.
- Furniture: 420 descriptions of digitally generated pictures of household furniture.

We converted the XML files of the corpus into predicate calculus, to construct referent cases. For a baseline, we used an exact match algorithm. In the first experiment, we used the TUNA corpus to test whether it could resolve correct descriptions intended to be uniquely identified. Here are the results:

	Exact Accuracy		**Accuracy w/ Ambiguity**	
	Ours	**Baseline**	**Ours**	**Baseline**
People	0.94	0.92	0.99	0.95
Furniture	0.80	0.78	0.98	0.92

Errors in the baseline system occurred due to a combination of annotator error, errors by the description author, and reliance on attributes that were not annotated in the corpus. The accuracy with ambiguity scores an answer as correct when multiple answers are returned, and the correct answer is included in that set (i.e. like hits@N). Here is an analysis of the kinds of errors that were made by the two systems:

	People		**Furniture**	
	Ours	**Baseline**	**Ours**	**Baseline**
Exact Retrieval	324	319	333	326
Ambiguous Retrieval	20	11	77	59
Incorrect Retrieval	2	2	7	6
No Retrieval	0	14	0	26

In the second experiment, we permuted the descriptions to test our system's robustness to noise. We performed three types of edit operations:

1. Insert incorrect attribute, e.g. `tunaColor-Green` when the target is a red chair.
2. Substitute a correct attribute with an incorrect one, e.g. `tunaAge-Young` with `tunaAge-Old`.
3. Delete an attribute.

[1] Gatt, A., Van Der Sluis, I., & Van Deemter, K. (2007, June). Evaluating algorithms for the generation of referring expressions using a balanced corpus. In *Proceedings of the Eleventh European Workshop on Natural Language Generation* (pp. 49-56). Association for Computational Linguistics.

The first two cases simulate near-misses, and the third simulates underspecification. We used between one and three edits. The baseline degrades rapidly, while analogical reference resolution is much more robust. These results were published in *Advances in Cognitive Systems* (Nakos et al. 2019).

We also extended our model to handle another important aspect of reference resolution: *perspective taking*. In dialogue, there is often a mismatch between what the speaker knows and what the hearer knows. This can lead to confusion. Psychological evidence shows that hearers use a two-stage process to take the speaker's perspective into account. The first stage makes a fast, initial judgment that only uses the hearer's knowledge. This is followed up by a slower process that corrects the initial judgment to take the speaker's perspective into account. The advantage of this two-stage process is that it avoids the need to model the speaker's knowledge explicitly, instead deriving it from the context as needed.

Our model uses analogical retrieval, as outlined above, for the first stage and a combination of theory of mind inference, suppression, and re-representation for the second. Theory of mind inference checks whether the retrieved entity is known to the speaker. For simple visual scenes, the check reduces to whether the speaker can see the entity, but more elaborate co-presence heuristics are possible. If the entity is determined to be unknown to the speaker, it is suppressed by removing it from the case library. The remaining cases are then re-represented to take the modified context into account. For example, in a scene with three rings arranged vertically, suppressing the bottom ring would make the middle ring into the new bottom one.

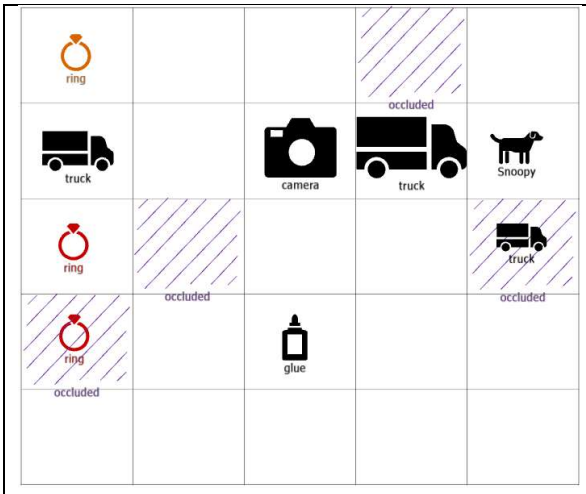
This last step provides a basis for handling adjectives whose interpretation depends on the entire visual scene. Words like "left" and "small" are interpreted in relation to a set of objects, not just the one being described. This makes it necessary to generate these attributes separately from underlying relations like `leftOf` and `biggerThan` and to re-generate them whenever the set of relevant objects changes. This allows our model to adapt its view of the world when dealing with a changing environment or taking the speaker's perspective.

We tested our model on a small set of stimuli used in a psychological experiment on perspective taking[1]. The experiment placed two people, a director and a subject, on opposite sides of a vertical grid containing various everyday objects. Some of the cells in the grid were occluded on one side so the director was unable to see the objects they contained. The subject was able to see all objects and was aware of which ones were hidden from the director. In spite of this awareness, subjects initially interpreted the director's object descriptions from their own perspective, picking a hidden object before correcting to an object the description applied to in the common ground.

We encoded the scenes from the experiment using CogSketch. The sketches were created manually and were annotated with information about the type, color, and size of the objects, as well as their occlusion status. These sketches were then used to automatically generate cases for the objects in the scene, using the process described above to fill in attributes like "left" or

[1] Epley, N., Morewedge, C. K., & Keysar, B. (2004). Perspective taking in children and adults: Equivalent egocentrism but differential correction. *Journal of Experimental Social Psychology*, 40, 760-768.

"small". We encoded the semantics of the description as a probe case, then used our model to interpret the description.



Example of a stimulus processed by the analogical reference resolution model. The model is emulating the behavior of the subject in a psychology experiment who can see all of the objects, and has to interpret what utterances by someone on the other side of the grid are referring to, given the subset of the objects that they can actually see.

Our model was able to interpret all of the descriptions from the experiment in the same way humans did. For descriptions where the speaker and the hearer's perspectives matched, our model settled on the correct referent using analogical retrieval and did not make any changes during the second stage of resolution. For descriptions where there was a relevant mismatch between the speaker and the hearer, our model followed the subjects' pattern of selecting the best match first before correcting to the best match visible to the speaker. These results were presented at the 2020 conference of the Cognitive Science Society (Nakos et al. 2020)

Unified QP Frames for Natural Language Semantics

One important role for qualitative representations is as a constituent of natural language semantics. The incremental nature of language means that information about situations and models typically arrives piece by piece, and must be assembled in order to create a qualitative model that can be reasoned with. Our prior research created distinct QP frame systems for instance-level and type-level qualitative models. This was inelegant as well as problematic, since the choice of constructing instance versus type level models ought to be made based on the text and task context, not a priori. Consequently, we have developed a new unified QP frame system which provides a layer that is agnostic with respect to whether the final description will be instance-level or type-level. This enables clues from the natural language analysis, task, and dialogue context to provide clues as to whether the final model constructed from the semantic analysis needs to be instance-level, type-level, or a mixture of both. In other words, there is a process of *model assembly* that produces a qualitative model for reasoning from the semantic analysis.

Consider for example a simple sentence: "Production depends on population." Notice this statement does not say whose production depends on what population, but there must be some entities for which these quantity types apply. Here is a unified QP frame description for the qualitative aspects of the semantic analysis:

```
Q1:
  quantityType: Production
```

```
Q2:
  quantityType: Population
Inf1:
  constrained: Q1
  constrainer: Q2
  influenceType: Qprop
  influenceSign: +
```

By itself, we cannot tell if this is a general piece of advice or a specific piece of advice targeted at a shared problem under discussion. Suppose the previous sentence in the dialogue was "You should increase production in Boston.", whose semantic analysis includes

```
Q0:
  quantityType: Production
  quantityEntity: Boston

(dqValue Q0
  (BeforeAfterMappingFn increase3) 1)
```

`dqValue` means a difference value, i.e. that the quantity `Q0` increased across the event increase3, the discourse variable representing the event referred to in the statement. Coreference resolution will merge `Q1` and `Q0`, hence this can be taken about a specific city. For an instance-level model, an appropriate entity must be found for the population quantity specification. A system this far along in understanding Freeciv will know that both cities and civilizations have population quantities, and in fact there is a qualitative proportionality between a city's production and its population. In that case, the model assembly process chooses the specific city, and hence produces

```
      (qprop (Production Boston)
             (Population Boston))
```

Thus this advice is focusing the system on something it already knows but might not have figured out is relevant in the current situation.

Suppose instead that the previous sentence were "Consider the production in a city." Then in that case the semantic analysis would include

```
Q3:
  quantityType: Production
  quantityEntity: City
```

and the coreference resolution process would merge Q3 with Q1. The entity is a type rather than an instance of a domain type, so the model assembler constructs a type-level qualitative statement, i.e.

```
(qprop+TypeType Production Population
                City City same)
```

for all cities, its production depends on its population.

Future work should build out the model assembly algorithms to handle all of qualitative process theory and test it against both instructional materials and interactive dialogues for advice.

Qualitative Reasoning for Decision-Making

Traditional models, such as reinforcement learning and decision theory, leave the framing of decision-making problems outside the formalism itself. Just as qualitative reasoning for science and engineering has provided a formal language for framing problems and reasoning about model formulation, a *qualitative theory of decision-making* should be able to formalize how to frame decision problems. This includes ways to specify what an agent should pay attention to, what choices does it have to make, how it should allocate resources according to its goals, and how it should evaluate progress. Our goal in this theoretical exploration is to develop a general theory, abstracting out from our previous and current work, and experiment with it using Freeciv. This section describes our work in this project on this area.

Strategic Thinking and Resources

One signature feature of our approach is to treat strategic thinking as a form of qualitative reasoning. Many strategies can be thought of as continuous processes, of the kind defined by qualitative process theory[1]. For example, exploration increases the amount of terrain which is known, expansion increases the number and quality of installations, and defense reduces vulnerability to attack. Even though these are implemented by the discrete actions of individual entities (sending units into the unknown, building cities, creating and positioning military units), these individual actions add up to achieving the desired effects, just as individual raindrops create puddles and floods. Thus we use qualitative process (QP) theory to represent strategies as processes.

A key factor in formalizing qualitative decision-making is developing a notion of resources. Resources include money, scientific progress and production, all of which can be treated as continuous properties, so QP theory also works well for these. Time itself is a resource, and we can slightly extend the notation of QP theory use its causal representations to handle time as well, e.g.

```
(qprop- (duration <construction>) (productivity <city>))
```

The definition of `qprop-` is in terms of a partial monotonic functional dependency, but typically both constrained quantities are fluents which are functions of time, which of course doesn't make sense for duration. This means the interpretation of causality when temporal properties are explicit parameters is limited to comparative statements, e.g. "if the productivity were higher, the duration to build it would be shorter." The estimation of duration via episodic memories described below in this report potentially provide the means of making quantitative estimations, e.g. if one city is more productive than another and the duration of the less productive city to produce an improvement is known, then the duration for a more productive city should be less, all else being equal. Space is also an important resource, and we believe that the qualitative spatial relationships that have already been developed by us and others will play an important role here.

[1] Forbus, K. (1984) Qualitative Process Theory. *Artificial Intelligence*.

We formalized the notion of qualitative resource models in terms of resources tied to events or actions. We follow the standard definitions of a consumable resource being something that is used up and a durable resource being something needed in an event or action but persists after its use. In baking, for example, the flour that went into a recipe is a consumable resource and the oven used is a durable resource. The costs of an action or event A in consumable resources consist of a set of quantities, one per consumable resource. For example, a representation of the typical costs of dinner at a fancy restaurant can be described as

```
(valueOf-Type (CostFn Money)
              DinnerNiceRestaurant
              (Dollar 50 100))
(valueOf-Type (CostFn Time)
              DinnerNiceRestaurant
              (Hour 2 6))
```

That is, events that are instances of the concept of having dinner at a nice restaurant cost between $50-$100 and they last between 2-6 hours. (We use the OpenCyc conventions that units, here `Dollar` and `Hour`, are represented via logical functions, which with two arguments indicates an interval and with one argument indicates a specific value.) Type-level qualitative models[1] are useful for reasoning abstractly, and enable the expression of general causal models, e.g.

```
(qprop+TypeType (CostFn Money)
                (CostFn Time)
   DinnerNiceRestaurant DinnerNiceRestaurant
   same)
```

That is, the longer a dinner at a nice restaurant takes, the more expensive it is likely to be. Numerical costs can be estimated via analogical generalization from experience as we have shown in this project (e.g. Hancock et al. 2018), but partial information is also useful. This can include ordinal relations (e.g. dinner at a fast food restaurant costs less than at a nice restaurant) and stratified values, including order of magnitude relations. For example, OpenCyc includes values of `Inexpensive` and `Expensive`, and while specific intervals are not associated with either value (compared to values like `SeveralHoursDuration`), they imply an ordinal relation.

Qualitative resource models concern establishing feasibility or relative desirability/cost, rather than optimality, because they are operating with partial information. For example, in deciding how to defend a city in Freeciv, the costs of different tactics can be represented by a resource model for each tactic. `DefendByReinforcement`, where a defending unit is moved into a city, has a cost in travel time, whereas `DefendByBuilding` depends on how long it takes to build a unit, and `DefendByBuying` involves a monetary cost, to speed production. This concept of resource models is described in a paper to appear in the 34th International Workshop on Qualitative Reasoning, later this month.

Our prior work on reasoning about goals included automatic identification of tradeoffs, which of course is an important factor in problem formulation. Another concept at this level are *gaps*, i.e.

[1] Hinrichs, T. and Forbus, K. (2012). Toward Higher-Order Qualitative Representations. Proceedings of the Twenty-sixth International Workshop on Qualitative Reasoning. Los Angeles CA, July.

the differences between the amount or type of a resource that one has and the amount or type that one needs (often determined relative to an opponent). For example, not having enough defense forces in a city can be considered as a material gap. When there are barbarians at the gates and reinforcements are far away, the problem is a time gap. If the best defender your civilization can produce is a chariot and your opponent has tanks, the problem is a technology gap. Technology gaps are often due to investment gaps, i.e. opponents are outspending you on research. Production gaps can make decisive differences in wars, as the US outproducing the Axis countries during the second world war. Future work should focus on formalizing methods for detecting, diagnosing, and repairing such gaps.

Costs, Benefits, and Functional Subsystems

We ended up formalizing costs in terms of type-level qualitative models, to better support higher-order qualitative reasoning. This required building dimensions into fluents. For example, the statement

```
(valueOf ((CostFn MonetaryValue) Ticket2) ((USDollarFn 9))
```

indicates that the cost of a movie ticket is $9.

This simplifies the expression of type-level qualitative models, which use implicit quantification, e.g.

```
(qprop-TypeType (QPQuantityFn FreedomOfAction)
                (NumberOfTechCapabilitiesFn EconomicSystem)
               Agent-Generic Agent-Generic equals)
```

which says that the freedom of action for a civilization can be improved by increasing the technological capabilities of its economic system.

Explicit representations of subsystems in a dynamic environment is one of the advances we made in this project. *Functional subsystems* represent a decomposition of the interlocking subsystems that constitute any complex system in aspects of the functioning of the overall system. In designing a smartphone, for instance, designers think in terms of power systems, software architecture, sensors, thermal dissipation, RF performance, and so on. These are not all separate components: RF performance and thermal dissipation, for example, are functional perspectives on the overall design that have a coherent set of goals and performance constraints, and hence are worth considering together as part of the decisions made in designing a phone. Similarly, countries are often thought of in terms of their economies, militaries, and other subsystems. In terms of qualitative decision-making, functional subsystems provide a means of expressing priorities. In Freeciv, for instance, rapid growth is crucial to be competitive with other civilizations, and to build a strong economic base that (as war breaks out) provides the ability to support a strong military.

Costs and benefits both have immediate and enduring aspects. Owning a car, for example, has the immediate cost of buying the car, and multiple enduring costs, including fuel and maintenance. Buying resources is an example of an immediate benefit, e.g. buying food in a restaurant means you can immediately eat it. Learning to cook, on the other hand, is an enduring

benefit, since presumably the skills you learn continue to provide value. The distinction between immediate and enduring benefits leads to an important distinction in types of investments. Some investments increase *capacity* by providing new resources that can be used in multiple ways. In Freeciv, for example, building military units or coinage provides increased capacity for defense and for building an economy. Other investments increase *capabilities* by providing new types of resources that provide additional freedom of action. Doing research in Freeciv, for instance, leads to new capabilities, e.g. the ability to create new kinds of units and buildings, which must then be produced by cities in order to add them to that civilization's capacity.

In Freeciv the distinction between capacity building and capability building comes down to choosing what cities are building and what research is being done, respectively. (Each city can be building something different, but research is only conducted at the level of the entire civilization.) To make these decisions requires linking a model of benefits to functional systems. Part of our qualitative model for Freeciv now includes such connections. This table shows the set of functional systems, an example of a unit or building that provides increased capacity, and a technology that can be researched to produce new capabilities for that system.

System	Unit/Building	Technology
Growth	Settler	Pottery
Economic	Marketplace	Currency
Research	Library	Writing
Awareness	Diplomat	Writing
Military	Warrior	Wheel
Transportation	Worker	Bridge Building

Technologies can of course impact multiple functional subsystems, e.g. Writing is a prerequisite to building both Libraries and Diplomats, and hence its discovery adds capability to both Research and Awareness subsystems.

Decision-making needs to be sensitive to priorities. The order of subsystems in the table above indicate a Companion's default priorities, e.g. growth first and foremost. But when war breaks out (an explicit state in the diplomatic subsystem of the simulation), improving the military should take priority, and our new decision algorithms change those priorities accordingly.

We evaluated these ideas by formulating algorithms for making decisions for what to build in a city and what research to do in a civilization. The qualitative decision methods were

- Production decisions: Use priorities over functional systems to select a subset of candidates and use ordinal properties over their benefits to select among those.
- Research decisions: Use priorities over functional systems to select which of the technologies that can currently be researched should be worked on.

Even in the space of qualitative decision algorithms, we have started very simply. Production decisions are not changing the evaluation of benefits based on details of the current situation, e.g. whether to focus on military units with higher mobility or higher defense strength, given the tactical picture. Research decisions are made locally, without lookahead. For baselines we used:

- Production decisions: Choose randomly from among the things that a city can build next.

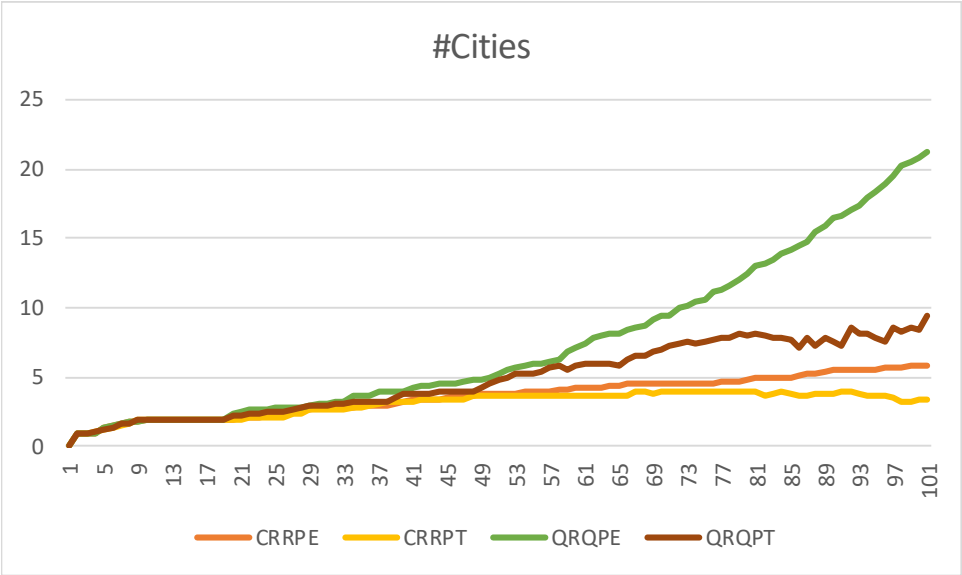
- Research decisions: Select the lowest-depth technology that can be researched. If there are several, choose based on alphabetical order.

The random choice baseline for production is stronger than it might seem, because the set of things that can be produced in the early game tends to be small. The closest technology heuristic is quite reasonable because the technologies in the technology tree have many dependencies, so to get very far, earlier technologies cannot be ignored.

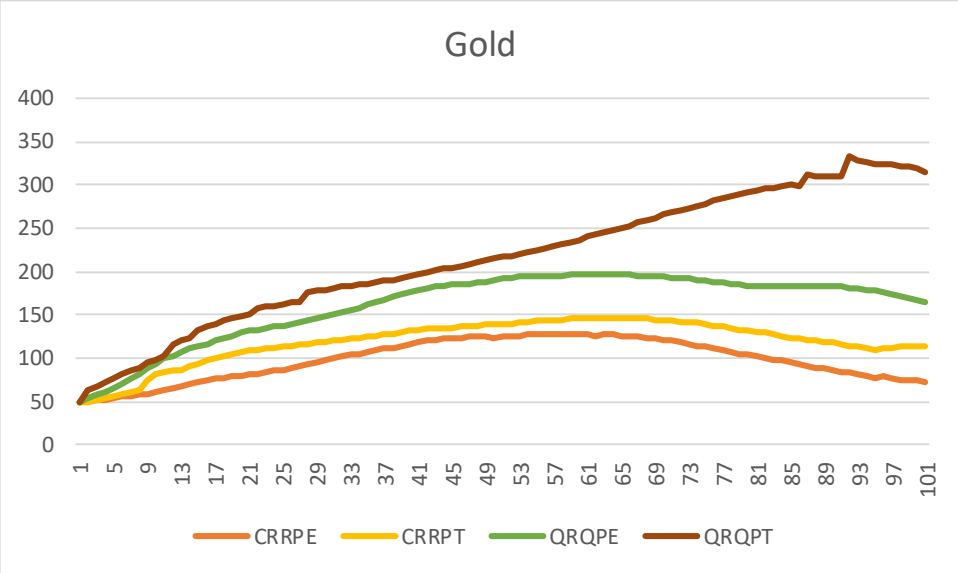
We examined behavior across two kinds of scenarios: (1) Empty maps: These test the ability of a strategy to support growth and expansion, factoring out warfare. (2) Combat maps: These worlds have four additional civilizations, with maps that ensure early and frequent combat.

Both types of maps use only 2,000 tiles (compared to 4,000 by default), with the land fraction set at 85% (the highest, default is 20%), so that all civilizations will start on the same continent. Huts and barbarians were turned off, since they provide random boons and banes that would distort the results. Ten maps of each type were used, and the same ten were used in all conditions. The simulation was run for 100 turns.

We looked at qualitative decision making versus the baselines. We track the number of cities and amount of gold produced, since to ultimately win the game, a civilization must outcompete others, even when wars are occurring. Here are the results for cities averaged across the ten maps of each type:



"CRRP" indicates closest research, random production decisions, while "QRQP" indicates qualitative research, qualitative production decisions. The "E" suffix indicates the data for empty maps, while "T" indicates combat-heavy maps. Note that the qualitative decision algorithms outperform the baselines on both types of maps. Similar performance is found for gold production, as shown below.



One problem with average data is that it can hide interesting variations within the samples. In fact, in combat-heavy maps, when using qualitative decision-making, the Companion failed to survive for the full 100 turns in three out of the ten maps. On the other hand, when using the baseline decision-making methods, the Companion always survived to the end. Inspecting the results suggests that there are two reasons for this. First, the larger civilizations that the QDM methods produce are more easily detected by the bots. Second, the baseline production decision method always produces a large number of military units, because that is all that can be built during the early parts of the game. This makes its cities very difficult to conquer, at the expense of the entire civilization never growing very much. While okay as a survival strategy, this is not a strategy that will lead to winning the game.

Notice that gold production rises and then falls over the course of play in all conditions. While some of the decline can be explained by early termination (i.e. QRQPT data), the decline in the empty maps can only be caused by maintenance costs rising faster than the civilization can support. This is the sort of phenomena that should be addressed by providing domain-independent models which the system can learn to apply to specific situations. Specifically, future work should extend tradeoff detection to detect tradeoffs in enduring costs versus benefits, so that experimentation techniques like those described elsewhere in this report can be used by systems to improve their own performance.

Another important direction for future work is being able to use natural language dialogue to extend the qualitative model and enable systems to discuss their reasoning. For example, some of the framing of decision problems involves translation from other representations into costs and benefits. The cost along any dimension is the sum of the costs of all the events constituting the execution of that alternative, plus the costs of any objects acquired/consumed during its execution. Decomposing such ideas into bite-sized chunks that can be communicated via language, in a domain-independent way, would help improve the ability of machines to receive instruction.

Learning from Episodic Memory

Organisms learn from experience in many ways. One component of learning from experience is recording what has happened in the world when actions are taken, a form of episodic memory. Our hypothesis was that distilling such experience via analogical generalization could enable systems to learn *analogical models* that provide expectations and guides to future actions. Moreover, the accuracy of such models can be monitored, to detect surprises and to help identify and prioritize learning goals.

We explored the use of analogical generalization over episodic memories in the Companion cognitive architecture to formulate models of the effects of actions in a complex dynamic world. Our research developed episodic memory techniques for measuring novelty and surprise and for estimating the duration of events. We also explored spatiotemporal representations in episodic memories, to improve strategic and tactical reasoning.

Distilling action models from episodic memories

How human episodic memory is organized is still an open question. Given the centrality of analogy in human cognition, as outlined in the Background section above, it seems reasonable that a common way of structuring episodic memories could be as cases, so that they can be accessed via analogical retrieval with more transferable knowledge constructed incrementally via generalization. The turn-based nature of Freeciv imposes a natural discrete structure on behavior: Each command that is executed can be thought of as an event. That event itself might lead to others: An attack, for instance, will lead to either the attacker or defender being destroyed, which is provided to the Companion as another event by the interface to the simulation.

We started by developing techniques for distilling models of actions via analogical generalization from a Companion observing human players. Learning from observation is an important aspect of apprenticeship, especially in early stages of learning a domain. This approach also enabled us to work with a set of recorded games, to experiment with different encoding strategies and to carry out sensitivity analyses more easily. For each action the person took, the Companion recorded information about the state of the world before and after the action, and used some general-purpose heuristics to attempt to explain immediate events in terms of the action (e.g. the destruction of a unit after an attack). For each occurrence of every action, a case consisting of this information was stored. Storage occurs via a SAGE generalization pool for each command (e.g., doMove, doIrrigate). The generalization pool for a command can be thought of as an analogy-derived model for what happens when that command is used. By letting the system watch replays from six different games, it built up models for 34 different commands from 4,200 cases.

Inspecting these generalization pools led to some interesting insights. First, the number of generalizations and outliers in a pool provides an indication of how well the action is understood. If there are many cases all forming a single generalization, then that command has

straightforward local consequences (e.g. doIrrigate)[1]. When there are multiple generalizations, comparing their structures can be illuminating: For example, in doResearch, the generalizations differ only in the number of requirements and opportunities, making them artifacts of the encoding strategy, which could be eliminated via re-representation. Thus properties of the generalization pools provide a signal about how encoding strategies might be improved.

Detecting Novelty and Surprise

Analogical generalization also provides a means of detecting and quantifying novelty and surprise. Novelty can be detected in two ways: Failure to retrieve a similar experience, and by analysis of candidate inferences indicating differences. When little is known, all is novel – surprise, we argue, occurs when a novel situation is experienced for a type of situation that was considered to already be well understood. The degree of surprise can be estimated based on the number of cases in the pool and frequency information for relationships within them that are computed for the generalizations: when there are many cases and highly certain outcomes, a new outcome can be more surprising. doMove provides an excellent example: It occurs frequently, so a dominant analogical model is quickly built up. But when a unit moves into a hut, there are five different things that might happen, leading initially to surprises.

To test these ideas, we formulated a concrete metric for surprise. Given a new example E of a command C, whose model consists of gpool(C), i.e. the generalization pool for that command, we define the novelty of E with respect to gpool(C) as

$1-NSIM_B(BestMapping(SME(E,MACFAC(E,gpool(C))))$

That is, the base-normalized numerical similarity score of the best mapping for the closest item retrieved from the generalization pool. If a case identical to (or isomorphic up to entity renaming) E is in gpool(C), then the numerical similarity will be 1, and E will have zero novelty. If nothing is retrieved, then the numerical similarity is taken to be zero, and hence E would have the maximum novelty of 1.

Not all novelty is significant. SAGE provides a natural definition of significance for novelty, since that can be taken as the dual of the decision that something is close enough for assimilation. That is, every gpool has an *assimilation threshold*, A_t, which ranges between zero and 1. To respect this threshold, if an example E would be assimilated under the current threshold, we let the novelty be equal to zero, since it is close enough to a prior experience or generalization to be assimilated.

Surprise, we have argued above, is novelty in a situation about which one has experience. We model this quantitatively by a simple rate equation:

$1-e^{-n/r}$

[1] These models only capture the immediate consequences of commands, e.g. not the changes to terrain that occur several moves later as a consequence of undertaking this action. Correctly inferring longer-term consequences of commands can potentially either be done by observation, described below, or by reading.

Where n = the number of examples that have been added to the generalization pool and r is a rate parameter, controlling how fast this parameter reaches its asymptote of 1.

We implemented this metric, and ran it over the 4,200 cases previously collected. The results are consistent with our hypotheses. For example, Figure 11 shows the graph of surprise values computed for the doMove command over a game. The 17th doMove command (which occurred at turn 26 in that game) leads to a small but nonzero level of surprise. This is the first occurrence of a unit entering a hut. The second occurrence of entering a hut is at turn 34 does not lead to a surprise: What is retrieved is the prior experience of entering a hut, and so it is assimilated. The subsequent occurrences of movements that enter huts are assimilated into that new generalization as well. The next move which enters a hut that causes a surprise involves more events and relationships than previous such events, since it involves finding a new technology, whereas the earlier ones involved the unit moving either getting killed or finding gold. The final surprise in this sequence is the first movement involving a ship moving on a deep ocean tile – such tiles have fewer attributes than shallow water tiles or land tiles, and this lack of information about the new location is a surprise.

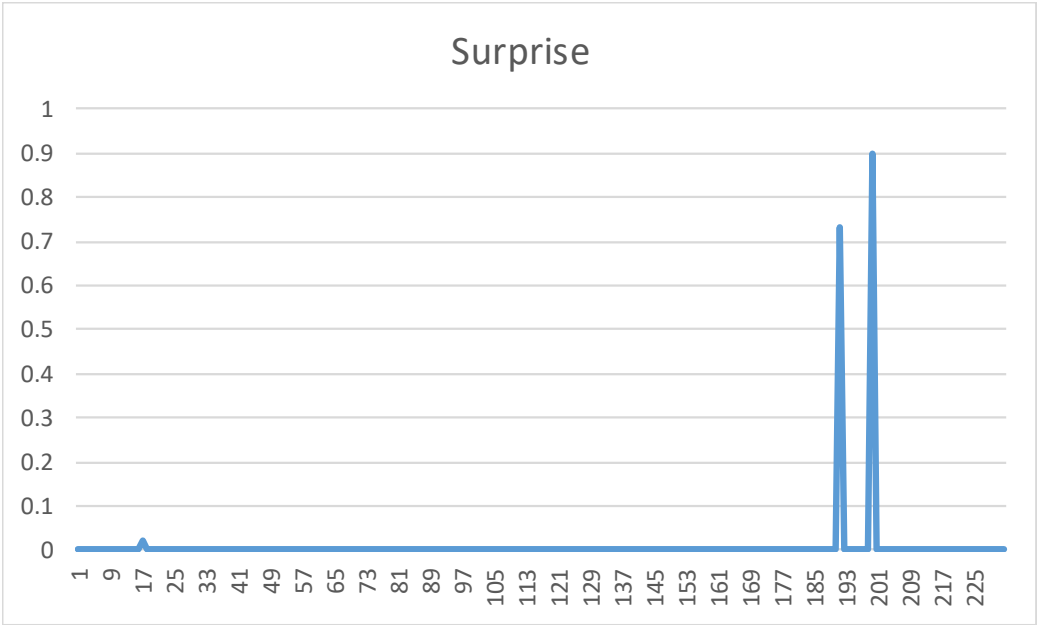


Figure 11: Surprise values for doMove over a game

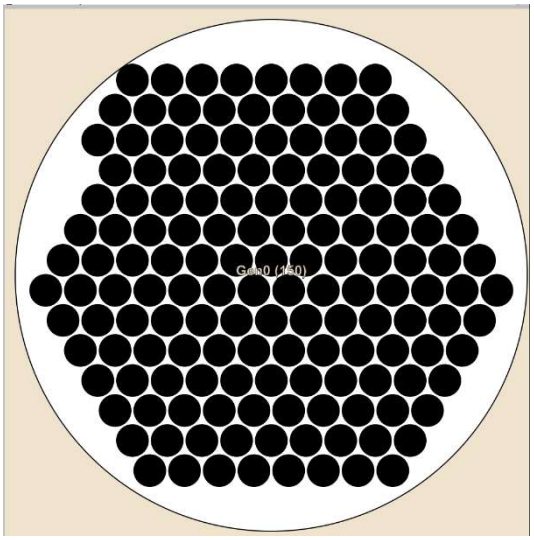
The evidence so far suggests that this surprise metric seems reasonable. Future work should include identifying the kinds of responses that a system should take to these surprises, developing a catalog of strategies and diagnostics for them, by analyzing both surprising events and generalizations formed from them. For instance, differences in the number of antecedent and consequent technologies in doResearch are not functionally important on their own, which suggests changing how such events are encoded in order to erase such differences. Differences in outcomes suggest building causal models to predict outcomes, to improve planning, e.g. doAttack. In some cases, outcomes will be best understood as stochastic (e.g. the consequences

of entering a hut). These strategies, which we suspect can be developed in domain-general ways, will be a useful part of the metaknowledge for systems that need to take charge of their own learning.

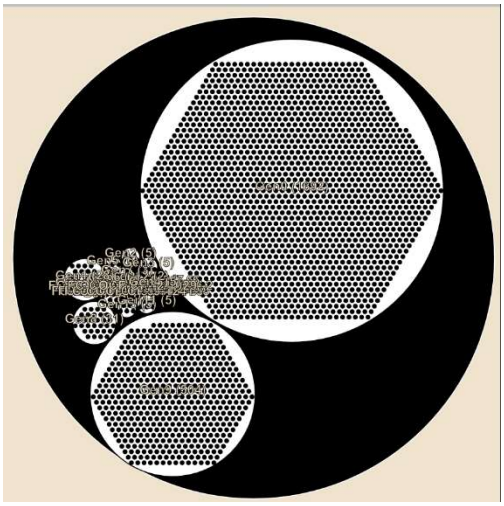
To summarize: our SAGE model of analogical generalization accumulates examples incrementally, merging very similar examples into generalizations that track the probability of each statement within them. Thus the generalization pool for a concept will have one or more generalizations constructed over time (multiple generalizations provide a representation for disjunctive concepts) and can also contain outlier examples (which may be truly exceptional cases, or just the seeds of new generalizations that will be formed as yet more examples come in). An example is novel to the degree that it is different from the most similar item (generalization or example) in the generalization pool, with novelty = 1.0 when it is so different that it retrieves nothing. If novelty occurs for a concept that the system has a lot of experience with, that can be considered a form of surprise, where the degree of surprise is modeled by a rate equation, so that novel examples when the system has less experience are less surprising, whereas a radically different example when the system has a lot of experience is very surprising. This is aimed at providing a signal that detects when something unusual happens, as a way of guiding attention and formulating learning goals.

To further explore our model of surprise, we performed a corpus analysis, using cases automatically constructed from a Companion watching six replayed Freeciv games. These cases concerned the immediate effects of primitive actions. This led to a set of 4,278 cases, involving 34 different types of commands. The distribution of commands was not even: doMove (i.e. move a unit from one tile to another) gave rise to 2,307 cases. We used this corpus to analyze two questions: (1) can generalizations yield insights? And (2) does this measure reflect intuitive properties of surprise?

Regarding the first question, there are several insights that can be generated from this measure. First, for many actions (e.g. doIrrigate, doBuildRoad), only a single generalization is created. This indicates that there aren't immediately detectable differences between different actions of that type, as the visualization below indicates. (In generalization pool diagrams, black circles indicate cases, and white circles indicate generalizations within that generalization pool. Black circles inside a white circle indicate the examples that went into constructing that generalization.)



This shows the generalization pool for doBuildRailroad. Each black dot is an example, and the fact that all of them are within the same white circle means that there is just one generalization. On the other hand, doMove has several generalizations:



Entering a hut is one source of variations among the immediate outcomes of doMove, since different things can happen. The statistics gathered by SAGE make it straightforward to calculate the probabilities of these outcomes:

Outcome of entering a hut	P
Gold Found	0.38
Technology Found	0.23
Unit joins your civ	0.23
City joins your civ	0.08
Killed by Barbarians	0.08

Such information is potentially useful in evaluating tradeoffs in plans.

To evaluate the second question, whether or not the measure reflects intuitive properties of surprise, we performed several analyses on the corpus of examples. First, surprises are rare, as indicated by the analysis of the six games in the corpus independently:

Game	#doMove	#Surprises	%
fc-sc1	101	2	1.98%
fc-sc2	296	9	3.04%
fc-sc3	232	3	1.29%
fc-sc4	280	4	1.43%
fc-sc5	996	12	1.20%
fc-sc6	397	10	2.52%

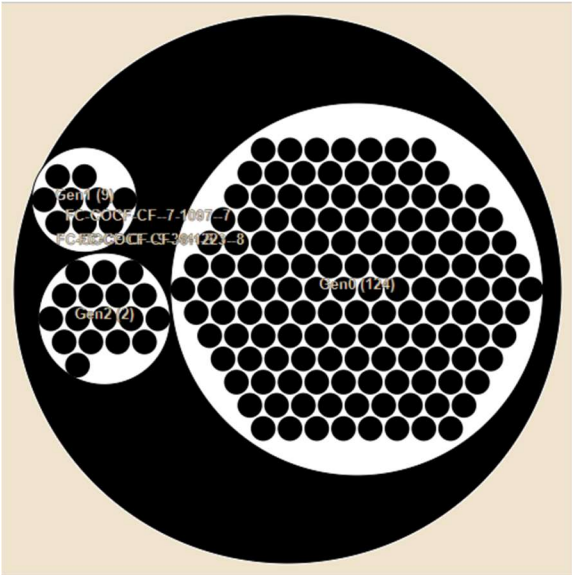
With cumulative experience, there are only 22 surprises across the entire set (0.96%). The first occurrence of a novel event generates surprise, but subsequent occurrences are typically subsumed into a new generalization.

While these results are encouraging, there remain open issues. One weakness in the current model is that novelty in the representation is not always novelty in the world – the first time a new enemy unit is revealed when moving is a surprise, but so is the first time two enemy units are revealed. A difference in number instead of kind might better be captured via a representation that reifies enemy units into a set and includes the cardinality of that set. Having the system automatically detect such re-representation opportunities and adapt its encoding strategy accordingly seems worth exploring. Nonetheless, we plan in future work to integrate the novelty and surprise metrics into the architecture, so that they can serve as control signals for attention and learning.

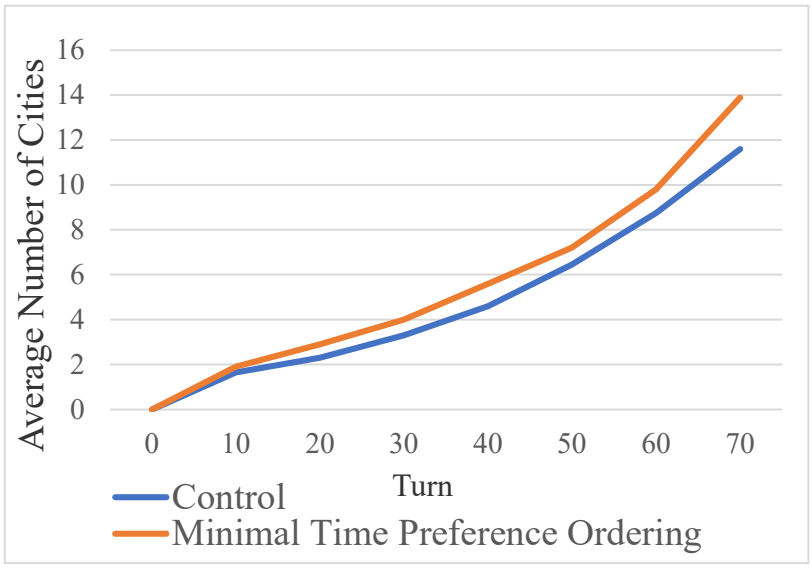
Estimating Action Durations from Episodic Memories

What else can be learned from episodic memories of actions? Understanding durations of actions is important for evaluating alternate plans, e.g. if the barbarians are already at the gate, a plan that can be executed quickly is more valuable than one requiring considerable lead time. Another example is assigning Worker units in Freeciv. Workers can be used to improve terrain, thereby providing benefits for nearby cities. Learning that irrigating grassland versus plains takes different amounts of time enables an agent to better plan how to use its workers. Consequently, we developed a technique for estimating the duration of actions from episodic memories via analogical generalization.

To estimate durations, the system must build cases that include not just the outcome, but how long it took. These cases are accumulated in SAGE generalization pools, one pool per type of action. For example, here is a depiction of the generalization pool for `doIrrigate`:



There are three generalizations in this pool, one much larger than the others. For each generalization, least-squares regression is used to construct a function for estimating duration, in terms of the relevant properties (e.g. type of agent, type of terrain, other binary relationships). When evaluating a potential action, the Companion builds a case corresponding to the start of the proposed action, and then uses analogical retrieval to find the closest generalization to it. The duration estimator for that generalization is then used to compute a time estimate for that action. This information is used in making choices about what actions to pursue, preferring to select the action that takes the least time. To show the value of this approach, we compare it with a baseline that doesn't take temporal cost of actions into account, using ten different Freeciv maps played for 70 turns each. Here are the results:

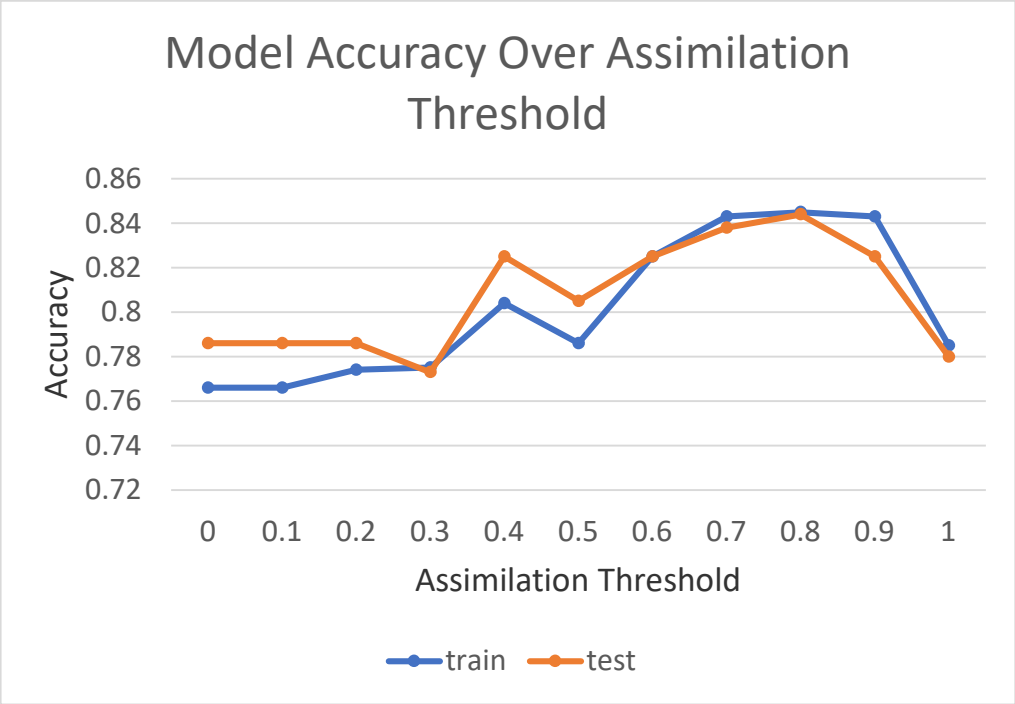


There were 16.5% more cities built on average when using minimal time preferential ordering, which is a statistically significant difference ($p < 0.0397$). This work was reported on at ACS 2018 (Hancock et al., 2018).

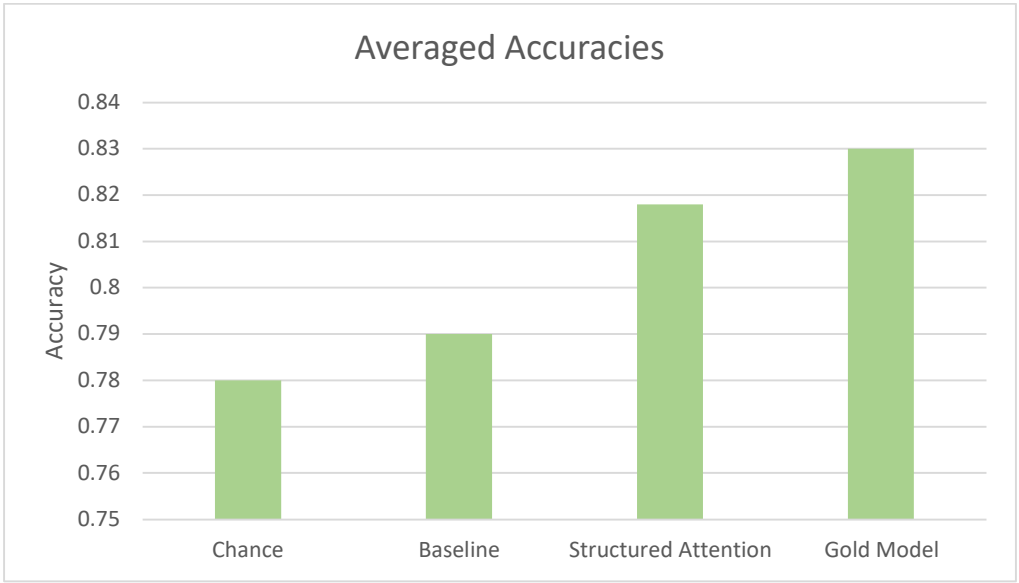
In prior work, we have argued that there is a trajectory of model development, starting with cases with incremental formation of generalizations and, when warranted, the construction of explicit rules[1]. This process, like all analogical reasoning, relies on good encoding strategies, to make explicit important information and suppress irrelevant details. To tackle the encoding question, we have examined using multiple representation schemes to improve analogical learning[2]; treating each representation scheme as a unitary whole. But what if some properties are more valuable, in information-theoretic terms? Can we use analogical generalization to focus on what relationships to encode, and can doing so lead to the production of useful rules?

To explore this, we looked at learning rules to estimate the outcome of combat in Freeciv. The idea is to compute both information gain for generalization pools and for individual properties within it. (By property, we mean matching relationships, e.g. that the attacker or defender is on a particular type of terrain, or the unit type of the attacker or defender, etc.) Recall that SAGE has an assimilation threshold which determines how close two items must be for them to be merged together. An assimilation threshold of 0.0 means assimilation will always occur, and an assimilation threshold of 1.0 means assimilation will only occur if the two descriptions are isomorphic up to entity names. As the assimilation threshold is increased, structural properties that are most responsible for cluster dissimilarity and that show higher mutual information between their values and cluster assignments are marked as salient and used to learn rules. The table below shows the accuracy of learned rules over different SAGE assimilation thresholds.

[1] Forbus, K. and Gentner, D. (1997). Qualitative mental models: Simulations or memories? *Proceedings of the Eleventh International Workshop on Qualitative Reasoning*, Cortona, Italy, June 3-6, pp. 97-104.
Friedman, S. E. and Forbus, K. D. (2010). An integrated systems approach to explanation-based conceptual change. *Proceedings of the 24th AAAI Conference on Artificial Intelligence*. Atlanta, GA.
[2] McLure, M. and Forbus, K. (2012). Encoding Strategies for Learning Geographical Concepts via Analogy. *Proceedings of the 26th International Workshop on Qualitative Reasoning*. Los Angeles, CA.



The table below shows the average accuracies of rules constructed by seven-fold cross validation, using 154 episodic memory cases involving combat. The baseline condition uses information gain from unstructured attributes (similarly to determining the root node of a decision tree), whereas the structured attention condition does entropy maximization (using the model from the assimilation threshold with the best train accuracy). The Gold model is a quantitative model based on the underlying game equations, but since combat in Freeciv is stochastic, even it does not provide perfect accuracy.



These results seem promising, so future work should include evaluating the robustness of this technique by applying it in other domains to test its generality.

Spatiotemporal Representations to Support Tactical and Strategic Reasoning

We expanded our representations of episodic memories to include *histories*, extended pieces of space time that carve up dynamic behaviors into meaningful regions[1]. That is, a Companion's memories of activity in a constructive dynamic domain (like Freeciv) will consist of collections of episodes of histories for the entities of the domain. For example, the footprint of a civilization grows as the civilization expands and can contract if things go badly militarily. Intervals of time over which the footprint is expanding, shrinking, or constant provide a relevant distinction for carving up time, because the causal factors governing the situation have changed.

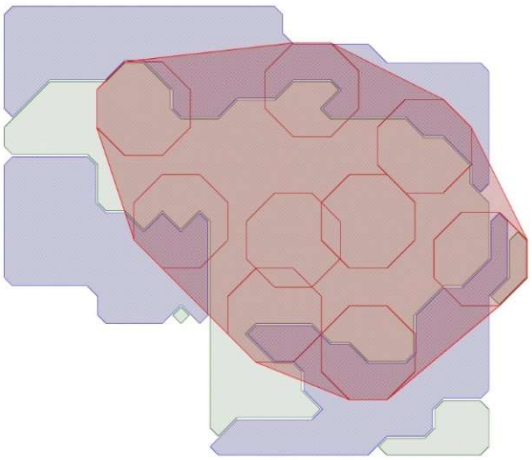
Our prior work showed that qualitative spatial representations automatically computed from terrain descriptions could be used to ground geospatial terms, like island and isthmus, in Freeciv maps[2]. Now we are exploring region-based qualitative spatial representations to provide the spatial information needed for strategic reasoning. Again, we are defining concepts in a general way, independent of the Freeciv game, and developing specialized encoding schemes that automatically compute instances of these general descriptions from Freeciv maps. This should lead to reasoning and learning that is transferrable to other domains.

In a competitive dynamic domain with a concept of terrain, the *footprint* of an organization constitutes the union of the spatial components of its parts. The footprint can be further divided into a notion of the *interior* of the terrain held, the *border* of a piece of terrain, and a *trigger region*. Valuable but vulnerable resources built up (in Freeciv, cities specializing in economic production or scientific research, for example) are naturally located in interiors, with borders requiring stronger defenses. Trigger regions are for controlling attention, i.e. entry of a competitor into that region should be monitored, so that plans can be generated (perhaps using retrieved tactics) for handling potential problems. The natural way to generate these descriptions from Freeciv is to use the network of cities as anchors and include their region of influence as part of the terrain. The convex hull of these regions provides a means of including the interior of a network of cities, with cities adjacent to the convex hull being frontier cities. Strategically, frontier cities are candidates for stronger defenses, resources permitting, to the degree to which there are threats that can come from the outside in their direction. For example, if Freedonia is to the west of Sylvania and to the east of El Dorado, then being at peace with El Dorado and at war with Sylvania suggests beefing up frontier cities on the east of Freedonia, and including more sentries in the eastern trigger region.

---

[1] Forbus, K. (2019). *Qualitative Representations: How People Reason and Learn about the Continuous World*, MIT Press.

[2] McLure, M. and Forbus, K. (2012). Encoding Strategies for Learning Geographical Concepts via Analogy. *Proceedings of the 26th International Workshop on Qualitative Reasoning*. Los Angeles, CA.



Example of a civilization's footprint, as computed by CogSketch. Green indicates land, blue ocean. Each red octagon represents the area of influence of a city. Here there are two interior cities, the rest are border cities.

We built qualitative spatial encoding mechanisms for automatically deriving qualitative spatial regions from Freeciv maps. We used CogSketch, our computational model of high-level vision and sketch understanding, to carry out these computations. Relevant entities for a qualitative representation were encoded in a CogSketch sketch as glyphs, whose coordinates are drawn from the map coordinates in the game simulation. Thus the encoding processes, once they are in CogSketch, can be domain-independent. This also enabled spatial reasoning to be performed efficiently via CogSketch's visual routines. For example, when a unit from another civilization crosses into a trigger region, that is automatically detected by the change in the qualitative topological relationship (RCC8-DC, i.e. disconnected, when outside, and RCC8-TPP, i.e. inside, once inside). Changes in the footprint boundary are recorded, as part of a quantitative description. While other distinctions may prove relevant as well, the simplest temporal decomposition was based on intervals of time over which it is increasing, decreasing, or constant, as measured by a sliding window, since growth happens in spurts.

Changes in the qualitative state and the tactics being employed also mark temporal divisions in the histories for the entities participating in those descriptions. One example is when a civilization shifts from building cities to focusing on economic growth, i.e. the BuildGrow strategy[1]. Another example is change in diplomatic state, e.g. from a cease-fire to a state of war. The prosecution of a war will be an episode, which consists of episodes defined by instantiations of strategies and tactics (e.g. take back territory, which in turn can be done by attacking cities or besieging them).

These episodes are both recorded as specific memories about a game, but also used to learn models of aspects of the game by analogical generalization, using SAGE, as we have done with primitive actions. The memories of specific games provide a more inspectable foundation for after-action reviews, where a trainer works with an apprentice to examine what it did and why, and make suggestions for improvement.

As a further test of the utility of our history-based episodic memory representation, we used it to learn production decisions for Freeciv cities. In addition to the previously discussed civilization footprints, we introduced two new footprints: city and unit footprints. These spatial entities were incorporated into larger episodes that describe qualitative change local to the city making the production decision.

Cities must take many factors into account when deciding what to build. In Freeciv, these factors might include whether a civilization is at war, whether or not there are enemies at the gate, how many defenders a city has inside its walls, how much gold a city has in its treasury, as well as the reserves in the national treasury, just to name a few.

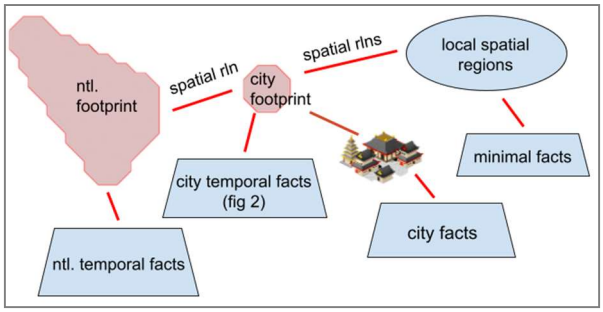


Figure 12: Anatomy of an episode. Local spatial regions may include continent blobs and other footprints.

We introduced *Qualitative SpatioTemporal Episodic Memory* that carves up continuous properties into local qualitative regions of space and time. One example is the period of growth or decline of a city. If the city size is currently monotonically non-decreasing, then this interval is represented in memory as a primitive. Using Allen Interval Algebra relations, it is then related to other quantity intervals that are temporally local (non-disjoint). These representations not only

[1] Hinrichs, T., and Forbus, K. (2015). Qualitative models for strategic planning. *Proceedings of the 3rd Annual Conference on Advances in Cognitive Systems*.

describe the immediate state of the world (gold is decreasing, enemies are approaching), but also

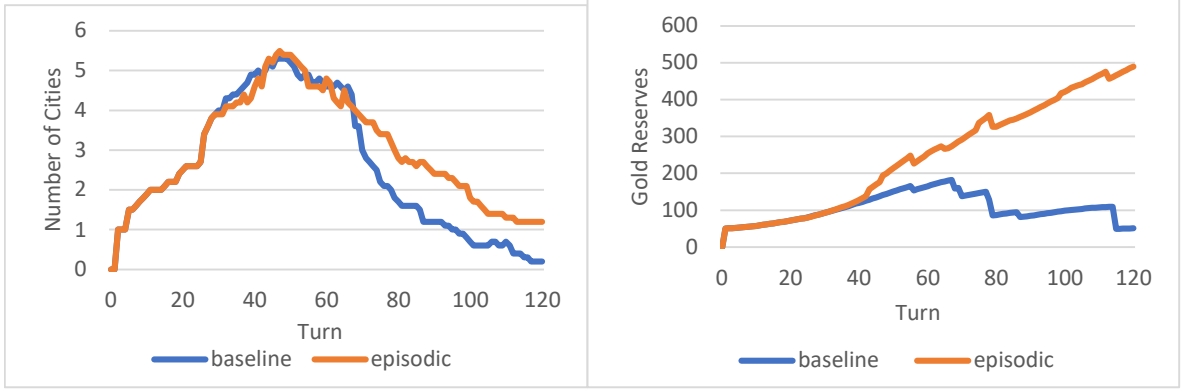


Figure13: Number of cities and gold reserves over time for two experimental conditions, averaged over ten games.

support describing what just happened, e.g. the national gold reserves were recently decreasing.

These episodic representations can help an agent learn production decisions from an expert. The idea is to make the same decisions as the expert in similar spatiotemporal contexts. To test this hypothesis, ten Freeciv games were played by an expert until turn 100, or the player was eliminated. Each time a production decision was made, a case was gathered, producing 460 cases overall. These 460 cases were generalized in SAGE and made available to the experimental agent.

The charts in Figure 13 show the number of cities and gold reserves over time for the baseline and experimental agents. The baseline agent makes investment decisions based on continuous quantities (e.g. should I increase the number of defenders in this city) and solely considers intrinsic properties i.e. properties shared by all members of their class. Extrinsic properties such as location, spatial configuration, and move points are therefore not considered. By contrast, the experimental agent explicitly represents and learns these types of continuous extrinsic properties. Both city count and gold reserves showed an improvement, with $p<.05$. Thus this representation also is useful for making production decisions.

Analogical Learning of Spatial Concepts

This was a line of investigation that we wrapped up from the previous project, as part of Matt McLure finishing his Ph.D. thesis. We used the TU Berlin sketch corpus to determine how analogical generalization performs in learning more complex spatial concepts. This research led to several innovations:

- The *edge-cycle* representation, which is a more abstract and concise representation of boundaries than edges.
- Automatic detection of *near-miss* examples via analogical retrieval from neighboring concepts, which enables automatic formulation of better representations for distinguishing close concepts. Winston's original notion of near-miss example required a

human teacher to identify a near miss based on full knowledge of the system's existing concept, and only allowed one difference. McLure's approach automatically identifies and uses near-miss examples without human intervention.

- An extension of the idea of support vector machines with the kernel being analogical mapping, which provides improvements over both analogical generalization and near-miss learning.

All but the last result were described in papers produced by the previous project, and McLure's PhD thesis (McLure 2019) describes them and describes the analogical support vector machines idea as well.

Honors and Awards

2017: Forbus was the inaugural recipient of the Herbert A. Simon Prize for Advances in Cognitive Systems

2020: Forbus was elected as a Fellow in the American Association for the Advancement of Science.

Publications

Forbus, K. (2019) *Qualitative Representations: How People Reason and Learn about the Continuous World.* MIT Press.

Forbus, K. D., Liang, C. and Rabkina, I. (2017). Representation and Computation in Cognitive Models. Topics in Cognitive Science. doi:10.1111/tops.12277.

Forbus, K. & Hinrichs, T. (2017). Analogy and Qualitative Representations in the Companion Cognitive Architecture. *AI Magazine* 38(4):34-42.

Hinrichs, T. & Forbus, K. (2017). Towards a Comprehensive Standard Model of Human-like Minds. *AAAI Fall Symposium.*

Forbus, K., & Hinrichs, T. (2018). Qualitative Reasoning for Decision-Making: A Preliminary Report. *Proceedings of the 31st International Workshop on Qualitative Reasoning*, Stockholm.

Forbus, K. & Hinrichs, T. (2019) Qualitative Reasoning about Investment Decisions. *Proceedings of the 32nd International Workshop on Qualitative Reasoning*, Macau.

Hancock, W., Forbus, K. & Hinrichs T. (2018) Optimizing Strategic Game Planning via Combining Analogical Learning with Function Learning. *Proceedings of the 6th Annual Conference on Advances in Cognitive Systems*, Stanford, CA.

Hancock, W. & Forbus, K. (2021) Qualitative Spatiotemporal Representations of Episodic Memory for Strategic Reasoning. *Proceedings of the 34th International Workshop on Qualitative Reasoning*, Online.

Hinrichs, T. & Forbus, K. (2019a) How Qualitative Models can Improve Learning by

Experimentation. *Proceedings of the 32nd International Workshop on Qualitative Reasoning*, Macau.

Hinrichs, T. and Forbus, K. (2019b). Experimentation in a Model-Based Game. In *Proceedings of the Seventh Annual Conference on Advances in Cognitive Systems*. Cambridge, MA.

McLure, M. (2019) Toward Automated Sketching Collaborators: An Analogical Route. Ph.D. Dissertation, Computer Science, Northwestern University.

Nakos, C., Rabkina, I., & Forbus, K. (2019) An Analogical Account of Reference Resolution. *Proceedings of the 7th Annual Conference on Advances in Cognitive Systems*, Cambridge, MA.

Nakos, C., Rabkina, I., Hill, S., & Forbus, K. (2020) Corrective Processes in Modeling Reference Resolution. *Proceedings of the 42nd Annual Conference of the Cognitive Science Society*, Online, July 2020.