Advances in Cognitive Systems X (20XX) 1–6 Submitted X/20XX; published X/20XX

# Experimentation in a Model-Based Game

**Thomas R. Hinrichs** T-HINRICHS@NORTHWESTERN.EDU
**Kenneth D. Forbus** FORBUS@NORTHWESTERN.EDU
Department of Computing Science, Northwestern University, Evanston, IL 60208 USA

## Abstract

One challenge for building software organisms is to support more autonomous, self-directed learning, rather than learning from annotated data or blindly exploring state spaces. We present a mechanism for learning a simple game given a qualitative model that provides partial information about how actions and quantities influence each other and about which goals trade off with each other. That information lets the learner progressively rule out unproductive actions based on qualitative state descriptions of the current situation and to experimentally adjust the relative importance of competing goals. We show that this amounts to operationalizing a qualitative model into a quantitative prescriptive model, which can lead to rapid improvement in performance on a game. In our experiments with a simple Human Resources Management game, it learned to win after one trial and continued to improve its score and reduced the number of actions needed to win throughout the next nine trials.

## 1. Introduction

Any human-like model of learning should account for the role of prior knowledge. When we learn a new task, we do not start from a blank slate, but, rather, expectations and beliefs guide actions and explanations to permit learning from far fewer trials than is the norm for today's statistical induction methods. We refer to this as *data efficiency*. One way that knowledge can guide learning is through self-directed experimentation. We pose questions to ourselves and take actions to winnow down uncertainty and triangulate on ever more accurate models. Knowledge about the domain can help pose questions that best refine these models as well as guide credit assignment.

Another way to achieve data efficiency is to support a more general notion of state. A learned action policy need not map from concrete primitive states to ground primitive actions, but may comprise abstract states and constraints on them that map to generalized actions. In this way, learning becomes a progressive refinement of states and actions that can stop as soon as performance plateaus, rather than exhaustively searching through primitive states. We bring together these ideas, experimentation, and reinforcement learning, using a qualitative model (Forbus, 2019) as the prior domain knowledge. We show how such a qualitative model can support self-directed experiments at a high level by exploring quantitative tradeoffs among competing goals. We also show how the same qualitative model can guide credit assignment to rule out ineffective action policies. In our approach, qualitative state representations further serve as antecedent conditions on learned action policy rules.

Previous work in active learning and experimentation has focused on supervised learning for classification tasks (e.g., Angluin, 1988) or domain theory acquisition and refinement (e.g., Gil,

1994). These approaches both result in efficient learners, but our approach differs in the nature of the prior knowledge available to the learner, the necessity for and means of credit assignment, and encoding learned knowledge as an action policy. Research on reinforcement learning has emphasized bottom-up statistical methods that assume no prior knowledge, at the cost of many learning trials (Sutton & Barto, 2018). While our learning mechanism is also unsupervised, it leverages a qualitative domain model to support efficient learning. We believe this will ultimately enable a continuum of approaches from highly interactive apprentice-like learning to fully autonomous experimentation.

In this paper, we describe a system that learns to play a simple game given a qualitative model of its mechanics. In the next section, we describe the domain, the Human Resources Manager game, and its design rationale. In Section 3, we describe how to play the game using a qualitative model and goal network. Section 4 presents the learning mechanism, including credit assignment, experimentation, the representation of experimental controls, and learning goals. Section 5 presents the results of empirical experiments, Section 6 compares this to related work, and Section 7 presents conclusions and plans for future work.

## 2. The Problem Domain

Human Resources Manager (HRM) is a single-player game in which the objective is to manage a small printing company for 20 months without driving it into bankruptcy or ending with a negative cash flow. The player starts with $50,000 and a roster of three employees, then makes HR decisions about hiring, firing, training, promoting, and giving raises. Unhappy employees quit and former employees sue the company if they were fired improperly.

HRM was adapted from a 27-year old corporate training simulator (Feifer & Hinrichs, 1992). It is implemented via backchaining rules in a form similar to the Game Description Language (Genesereth & Thielscher, 2014). We chose HRM because it was simple to implement, it has a complex underlying mathematical model, and yet it factors out adversarial and stochastic complexities. This provides a simple testbed to explore ideas about autonomous experimentation by enabling the system to control quantities and actions. We make no claims for its entertainment or pedagogical value.

Negotiating tradeoffs is key in this game, as in most strategy games. Finding an effective compromise between competing demands is an abstract task that is a major constituent of learning strategies. One of our research goals is to discover how to acquire such strategic knowledge with the same basic mechanism as learning action-level policies. There are three main tradeoffs in HRM. First, the goal to reduce labor costs with a low headcount competes with the goal to maximize income. Second, the goal to keep employees happy with high salaries competes with keeping salaries low to minimize labor costs. Third, the goal to invest in employee training competes with keeping payroll costs down. Discovering quantitative compromises for these goals can be viewed as turning a qualitative model into a partly quantitative model.

## 3. The Game Player

Before describing the learning mechanism, it is helpful to first understand how the game player works on its own. The player initializes the game state, consisting of quantitative properties and relations of the simulated company. On each turn it queries for legal actions, selects one, and applies it, then computes the next state. Most actions are at the domain level and can be applied to

individual employees, such as giving a raise or evaluating them. These have immediate effects, so we refer to them as *synchronic*. There is a special *diachronic* operator, doNextTurn, that advances the simulated time by one month. This lets the player take any number of actions within a turn and then explicitly advance the time. This happens automatically when there are no more viable actions in a turn. The game is over when the query for a terminal state succeeds, at which point the score is computed.

Selecting good actions is what the system must learn. Instead of starting with a blank slate, as most reinforcement learning agents do, it has a qualitative model of the quantities in the game, the graph of their influences, and the qualitative effects of actions on quantities. For example, giving an employee a raise increases their salary, which in turn positively influences the employee's attitude and the company's labor costs. The learning problem is to figure out how to balance these competing factors and identify conditions for taking actions.

The qualitative model for HRM was produced manually by abstracting equations in the game's rules. Prior work has shown the feasibility of learning a qualitative model from demonstration (Hinrichs & Forbus 2012), but this was not the current research focus. The HRM model has 37 reified quantity types and 53 influences linking quantities, actions, and events. A quantity type may be instantiated for each employee or for the company itself.

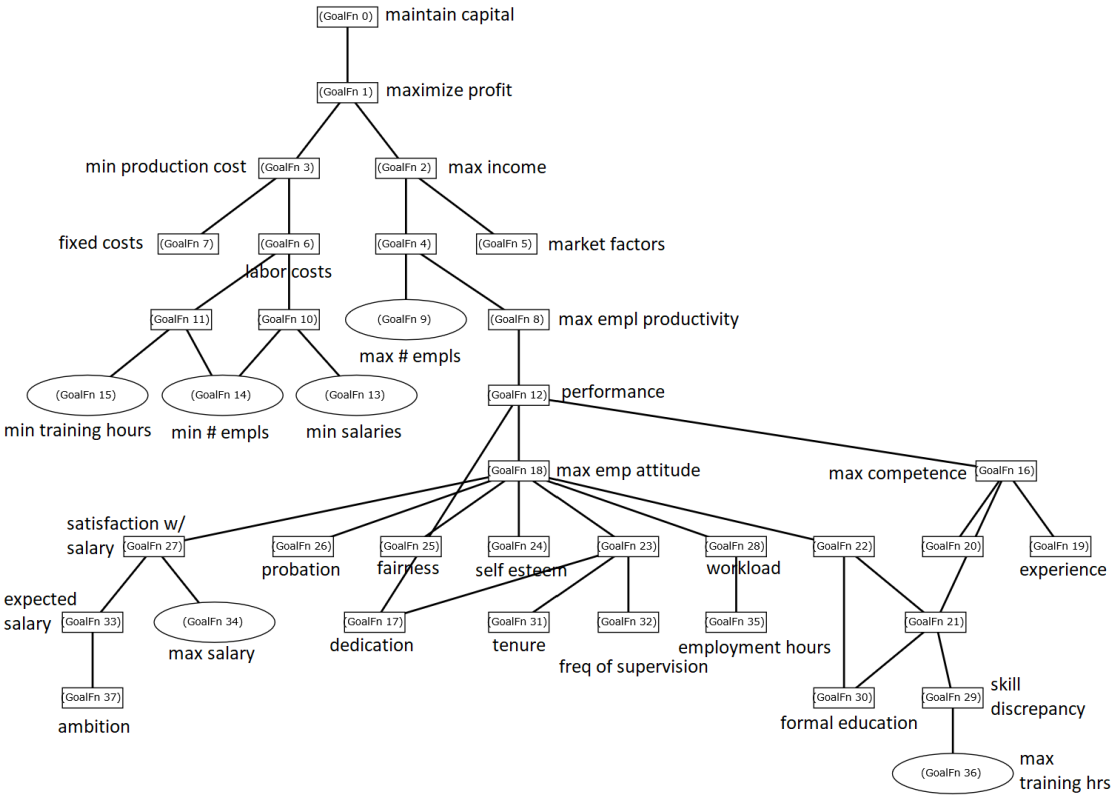


Figure 1: Goal network for HRM computed from the qualitative model.

Because the qualitative model ultimately connects intermediate quantities like salary to the top-level game goal, it is possible to automatically translate the quantity influences into subgoals. A static analysis routine walks the qualitative influences starting from the root goal quantities and reifies goals, as described in Hinrichs and Forbus (2016). Here, the goal types that are produced are all of the form 'maximize (or minimize) some quantity type'. As a side effect of building this goal network, static analysis detects tradeoffs by identifying quantity types that both positively and negatively influence the same quantity. Figure 1 shows the goal network for HRM, where the oval nodes indicate goals with direct tradeoffs.

We refer to a goal in this goal network as *operational* if there is a qualitative influence between some primitive action and the goal quantity. For instance, maximizing an employee's salary is operational because there is a qualitative dependence of employee salary on the action doGiveRaise. Higher-level goals, such as maximizing employees' attitudes, may be active but are not operational because there is no direct control over attitudes.

The reified goal network serves an additional purpose of keeping track of the relative *activations* of goals throughout the game. These roughly correspond to each goal's importance and thereby the proportional allocation of effort expended in pursuing it. Conceptually, if the top goal to win the game has 100% activation, then that activation is subdivided among its subgoals. By default, activation is allocated evenly, so that it serves as an informal proxy for importance relative to the top-level goal. Activation will be set to zero, however, if a goal type has no entities to which it applies. For example, a goal to maximize employee salaries will be inactive if there are no employees. Also, goal activation can be explicitly set by a meta-level planning action. This is how experimentation sets a tradeoff balance when exploring tradeoffs.

The effect of goal activation is to control the likelihood of picking actions that serve one goal over another. For goals that are pertinent to a single entity, such as the company, this results in stochastically picking an action or not, whereas for goals that apply to many entities, activation serves to divide actions by entity. For example, if the goal of maximizing salaries is only 20 percent, then only one fifth of employees should receive raises. We refer to this as an *action budget* for a type-level goal. The action budget ensures that no single action type monopolizes the available resources. Although it still allows raises to be given every turn, the action policy refinement learns to suppress this when the actions have no positive benefit on higher-level goals, as described later.

The pseudocode in Table 1 summarizes the action selection process. When the game player chooses an action to take, it steps through active, operational domain goals in decreasing order of activation. The system identifies action predicates that influence the goal quantity and queries for ground legal actions. If there are action policies or experimental conditions on the action predicate,

*Table 1*. Procedure for action selection with a qualitative model and an action policy

**foreach** domain *goal* in decreasing order of activation **do**
  **while** meets_action_budget(*goal*)
    *legal* ← legal_actions(*goal*)
    *acceptable* ← filter_by_action_policy(*legal*)
    *action* ← $\underset{a \in \text{acceptable}}{\text{argMin}}$ (goal_performance(entity($a$), $goal$))
    Take action
    Record before/after quantity changes
    Refine action policy

it filters the actions and selects the action whose entity argument is underperforming the most with respect to the goal (hence, argMin with respect to goal_performance). For example, only the most underpaid employees should receive raises. Finally, it takes the action in the game and records the quantity changes as it computes the next state. Any expectation violations here are passed to credit assignment to construct or refine an action policy for the action predicate.

## 4. Learning Mechanism

Our primary objective is to devise a method that learns abstract lessons autonomously from as few trials as possible. To this end, experimentation helps to *curate* experience by strategically guiding exploration, while credit assignment extracts more powerful lessons from each trial. Supporting both of these is a qualitative model that guides both activities. What is learned are action policy rules that allow or prohibit an action based on a qualitative description of the game state. These rules are refined progressively with experience, both immediately after taking an action and retrospectively, after an event signifying a failure (typically game loss). In both cases, the qualitative model guides the identification of salient quantities whose values create or refine the initial qualitative state description.

### 4.1 Credit Assignment

Part of data-efficient learning is drawing more powerful or general conclusions from each trial. Credit assignment seeks to explain the underlying cause of a failure by reasoning about the chain of qualitative influences from an action to a manifestation of failure. When the player loses a game, it carries out a *post-mortem* analysis. It looks back in time to the most recent action that set it up to lose the game. In general, this is an arbitrarily hard problem, but the qualitative model provides strong guidance in reconstructing the causal trail back to poor decisions.

Post-mortem analysis examines the loss situation to identify the quantities contributing to the loss. For HRM, this is the company's capital (cash reserves) reaching zero. It then traces backward in time, looking for a change in the rate of change of capital until it reaches the turn in which some action must have influenced the company's capital. It searches the indirect (i.e., synchronic) influences on capital until it finds an action that negatively impacted the profit rate, such as giving a raise or firing somebody. The post-mortem uses the results of credit assignment to post learning goals to learn the conditions under which the action primitive should or shouldn't be applied, creates or refines the action policy for that action, augments the set of quantities to be recorded for future credit assignment, and schedules follow-up experiments to further refine the policy.

Another kind of credit assignment happens during the game, when the result of an action does not have the expected effect on downstream, high-level quantities. This does not mean that the action was incorrectly described but, rather, that in the particular quantitative state under which the action was taken, it affected multiple quantities that combined in a way that reduced the overall performance. For example, firing a good employee might reduce labor costs, but might decrease income more. This immediate credit assignment doesn't need to walk back through time, since it already knows which action was taken. The routine identifies salient quantities as those whose value changed when the action was taken and caused the undesirable downstream quantity change. With this information, it refines the action policy for the action using the same mechanism as post-mortem credit assignment.

### 4.2 Generalization

To prevent the same mistake from being made in similar circumstances, the agent constructs an action policy for that action. Whereas an action policy in most reinforcement learners maps directly from states to utilities, our learner instead acquires and progressively generalizes *constraints* on actions. In particular, an action policy rule conditions an action specification with a qualitative state. The action specification either requires or prohibits an action, which may itself be lifted or generalized. For example, a policy might prohibit promoting AmbitiousAlice when her performance is less than 20 and her attitude is less than 50. Such a rule would look like

```
(controlConditionLowerBound
  (LearnCondForActionFn doHRMPromote)
  (MostSpecificConditionFn doHRMPromote)
  (ruleOut (doHRMPromote AmbitiousAlice))),
```

where first argument is the learning goal, the second argument is a functional term denoting the name of a model fragment that defines a qualitative state, and the third term is the action specification. The model fragment, in turn, relates the quantity conditions:

```
(and (< (performance AmbitiousAlice) 20)
     (< (attitude AmbitiousAlice) 50)).[1]
```

As new failure or success instances are encountered, the ranges on quantities are extended and the arguments to the action specifications are lifted as necessary. We adapted this representation to support experimental controls and it has the additional benefit of being relatively concise and explainable.

### 4.3 Autonomous Experimentation

Autonomous or self-directed experimentation is the process by which the learner proposes and executes experiments to reduce uncertainty. There are two reasons for self-directed experimentation: to strategically curate experience and to simplify credit assignment. The system addresses the former by systematically varying experimental parameters and the latter by controlling other exogenous parameters to restrict possible causes of change. In addition, experiments are organized around explicit declarative learning goals as a way to be strategic about the exploration process. These learning goals specify two different kinds of manipulation: *action experiments* and *tradeoff experiments.*

An action experiment is created when a postmortem traces a failure to an action that either directly caused a game loss or caused a trend that ultimately led to the loss. The agent posts an action-condition learning goal to refine the conditions under which the action is advisable. It then schedules experiments to refine the conditions by exploring the region between the most specific state to rule out and the most general. In other words, it reduces the uncertainty by driving the qualitative state conditions in a manner similar to candidate elimination in version spaces (Mitchell, Utgoff & Banerjii, 1980).

Tradeoff experiments, on the other hand, attempt to explore higher-level decisions by controlling the relative activations of competing goals. For example, if the baseline allocations of activation for competing goals are evenly divided, then a tradeoff learning goal will spawn two experimental trials, emphasizing first one goal, then the other by setting its activation to 75% vs.

[1] We have simplified the syntax here for the purposes of presentation.

25%. Subsequent experiments further extrapolate or interpolate the best performing allocation so far. These tradeoff experiments go even further toward simplifying credit assignment by suppressing all actions that cannot influence either of the competing goals. Consequently, this tradeoff learning mechanism can be thought of as an offline policy. Because tradeoff studies manipulate a single independent parameter (the ratio of two goal activations), no credit assignment is needed at all, and it identifies the best quantitative tradeoff by hill climbing based on the game score. One caveat is that this assumes tradeoffs themselves are independent of each other.

## 5. Empirical Evaluation

To evaluate our approach to learning game expertise, we ran trials under the two conditions. In the first study, we tested action learning by having it play autonomously through pure trial and error while honoring the goals and qualitative model. Our hypothesis was that the goal network would suggest plausible actions to take and the accumulation of action policy constraints would further refine the conditions under which they were attempted, yielding incremental performance improvement beyond merely surviving twenty turns. We measured the operating capital of the simulated company at the end of the last turn. In these trials, the agent learned to rule out actions that failed to have an immediate benefit as predicted by the qualitative model. It also learned from post-mortem analysis to rule out actions that had a long-term negative effect leading to a loss of the game. Initially, performance was spectacularly bad. Because every action in the game serves some goal, it micromanaged and tried to pursue every action as often as possible. In some cases, it tried firing everybody in the first few turns, leaving the fixed costs to drive the company into bankruptcy shortly afterward.

Figure 1 shows the results of the first three trials and the tenth trial. Each chart shows the progression of the company-wide capital, income, and production cost over time. While the first trial ended with bankruptcy in turn 5, by the second trial, it had learned an action policy that ruled out firing employees in most conditions and had discovered that hiring more employees was the key to surviving past turn 20. Trials 3 through 10 continue to improve the final outcome by increasing the profitability of the company until it banks $240,000 by turn 20 in trial 10. This bore out our expectation that performance would continue to improve even after it technically learned to win the game.

In addition to the performance curves, the charts also present the actual sequence of actions and events that occur in the trial. We can see from this that it quickly stopped firing employees and learned to hire sooner in the game. Moreover, as it refined the action policies, it learned to play with a lighter touch, such that by trial 10, it achieved better performance with far fewer actions consisting of hiring additional employees, giving a few raises and evaluations, one promotion and one training course. So while the qualitative goal network suggests that every action serves some goal, the gradual refinement of action policy adds quantitative constraints on when it is effective to take those actions.

To be clear, this is a very simple game. It is deterministic and the game objective is not especially difficult to achieve. In fact, under the baseline conditions of taking no actions at all, the company only fails after 19 turns. However, the point of these experiments is to show how quickly it is able to improve given fairly minimal background knowledge.

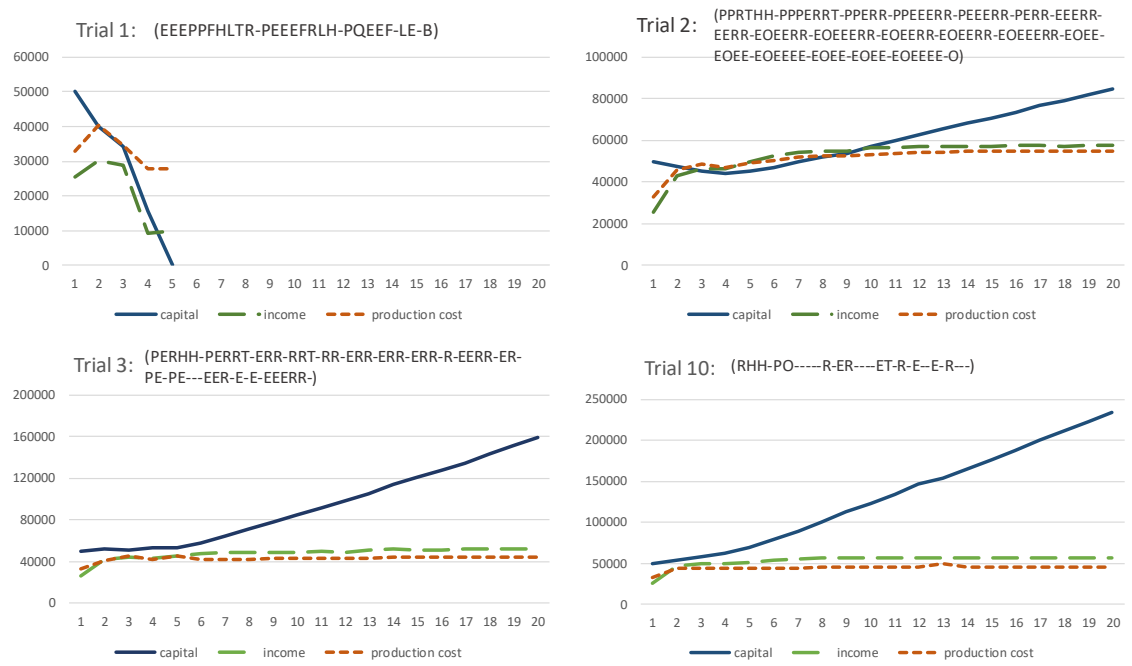


*Figure 1.* Action learning trials 1-3 & 10. The action abbreviation key is: **E**valuate, **P**romote, **H**ire, **F**ire, **R**aise, **T**rain, **L**awsuit, **O**verpaying, **B**ankruptcy.

Our second study examined tradeoff learning. Our hypothesis was that qualitative goal tradeoffs can provide a small number of parameters that can be explored, resulting in global improvements in behavior. Our metric was the same as for the action learning trials, and the results are shown in Figure 2. Here, because tradeoffs can be enumerated ahead of time, an initial set of six trials was scheduled to extrapolate tradeoff ratios in either direction from the baseline tradeoff allocation. In the first two trials, the agent explored the salary tradeoff by first setting the activation of the goal to minimize salaries at 50% vs. 0% for maximizing salaries. Of course, since there is no action to reduce salaries, this translates to never giving a raise. Moreover, since all other actions are suppressed in this offline policy, Trial 1 is equivalent to the baseline condition of taking no actions at all.

After this, the agent explored the tradeoff between reducing labor costs by omitting training (Trial 3) and increasing employee competence by training employees (Trial 4). As with the salary dimension, there is no 'untrain' action, so by not training anyone and suppressing all other actions, Trial 3 is equivalent to the baseline condition. Trial 4 did train an employee in the first turn, but the only evidence was a small spike in production cost, causing it to lose two turns earlier than in the baseline condition. In the final pair of trials, the agent explored the tradeoff between having fewer employees to reduce labor costs (Trial 5) and having more employees to increase production (Trial 6). The effect of reducing head count by firing approximately half the staff was swift and severe: labor costs dropped, but fixed costs stayed the same causing profits to nosedive, leading to bankruptcy in turn 6. Finally, in Trial 6, by hiring two additional employees at the beginning of the game, income (barely) exceeded production cost and the company remained profitable.

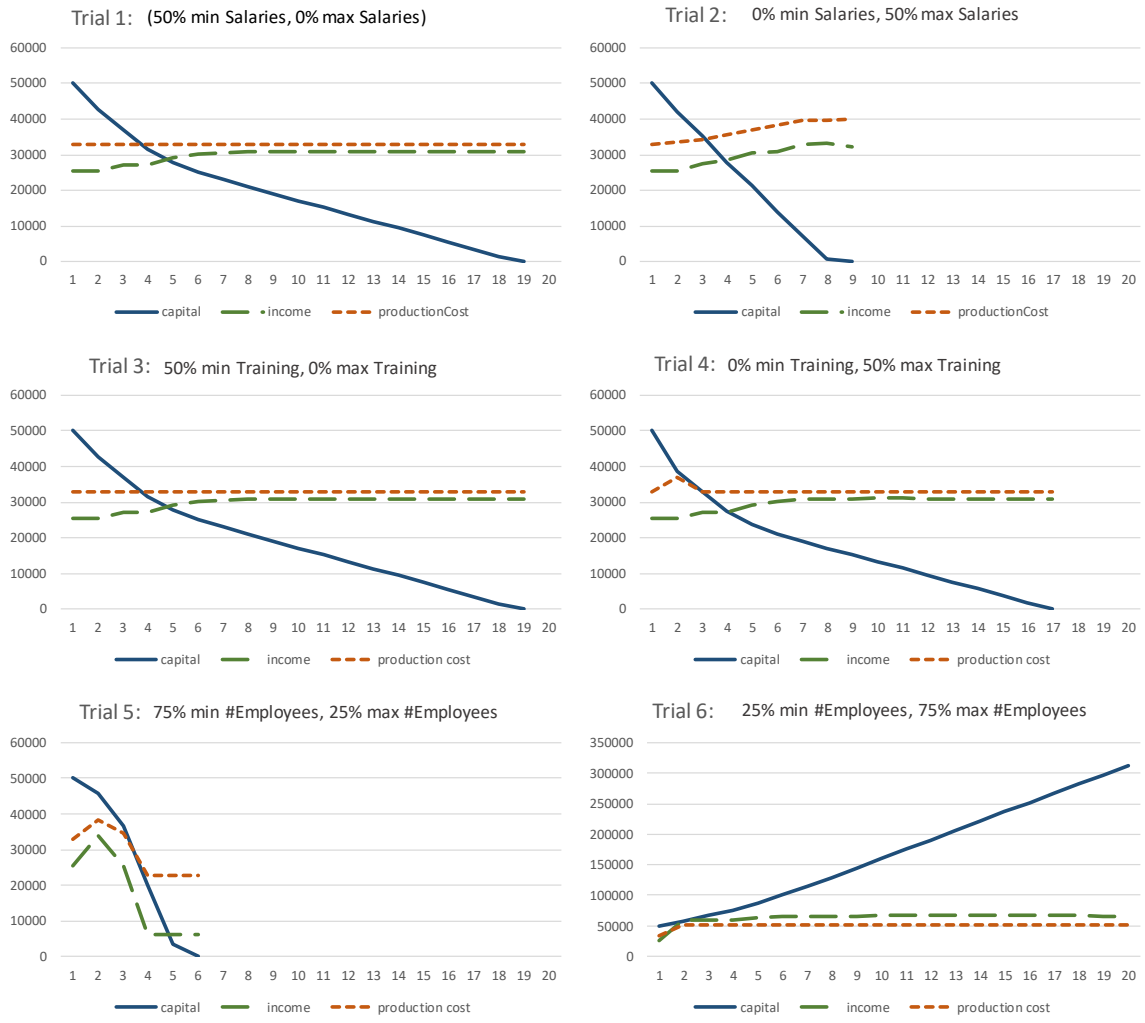


*Figure 2.* Tradeoff learning trials. Three pairs of trials explore the three domain tradeoffs by setting the relative activations of competing goals.

Because the tradeoff experiments suppressed all actions that were unrelated to the goal tradeoff, they were essentially *off-policy* learning (i.e., they did not exploit prior learning). Consequently, the trials do not form a learning curve and the results would have been the same if performed in any order. The purpose of imposing such experimental controls was to eliminate confounding factors that could hinder accurately determining the better tradeoff ratio. Ultimately, the tradeoff trials merely suggest of one way for a learning agent to experiment at a more abstract level than individual primitive operators. As currently implemented, the relative goal activations of competing goals are a coarse mechanism for controlling behavior and further refinement of the tradeoff ratios would not appreciably improve performance in this domain. Despite this, the system learned to win the game in six trials, which is data efficient by most standards.

## 6. Related Work

The approach described here derives from ideas in several areas, most notably automated experimentation and active learning, reinforcement learning, and qualitative modeling.

The main driver has been the desire for an agent to design its own experiments and pursue its own learning goals in ways that make use of available domain knowledge. This builds on work in *active learning* and *experimentation*. The primary difference is that active learning is a semi-supervised task in which an agent judiciously chooses queries to pose to an oracle to quickly learn to classify instances of a concept (Settles, 2012). Like active learning, our approach emphasizes data-efficient induction that uses prior knowledge to identify and prioritize knowledge gaps. Our action policy refinement mechanism has some precedent in the approach to hypothesis refinement as a version space search. Unlike active learning, we focus on unsupervised acquisition of tactical or strategic behavior in game playing, rather than on classification.

Experimentation is a more fully autonomous, unsupervised form of learning, in which the agent must design and run experiments on a syste, and validate or refute its own hypotheses. Part of this involves imposing experimental controls to minimize conflating factors and simplify credit assignment. Important early work in experimentation include operator refinement (Gil, 1994), which acquired domain knowledge about operator applicability. This used experimentation to identify and refine missing conditions and effects of planning operators that led to anomalous outcomes in execution. Like operator refinement, our system schedules and runs experiments to refine the conditions under which an operator can or should be applied. Unlike operator refinement, we are less concerned with repairing incomplete domain theories and are more focused on learning the advisability of different actions in different situations to improve behavior. We use a qualitative domain model to guide credit assignment and to concisely encode experimental controls. In addition, the model fragments used to specify the applicability of an operator are easily decomposed into sets of inequalities that effectively turn experimental design into a search in a parametric space. In our approach, an experiment generalizes or specializes a quantity condition while holding other conditions fixed.

More recent work tends to blur the distinction between active learning and experimentation, (e.g., Wang, Garrett, Kaelbling, & Lozano-Pérez, 2018; Konidaris, Kaelbling, & Lozano-Perez, 2018). These focus on model and representation learning rather than classification and involve active sampling of the environment rather than querying of an expert oracle. The objective in these efforts is to learn a model or symbolic representation from perceptual data with which an agent can plan. Because the learning agents are physical robots, data-efficient learning is critically important. At a high level, our approach is similar, except that our learner starts with a symbolic, qualitative model and learns to improve its performance on an abstract game task. The point of our experiments is to better understand the benefits of this initial knowledge endowment for learning.

Reinforcement learning is another paradigm from which we have borrowed liberally. This varies along several dimensions, including whether it is model free or model based, whether states are discrete or continuous, how and when exploration is carried out, what kind of feedback (reward or punishment) is provided, how it accounts for delayed effects, and whether actions are flat or hierarchical. The technique we have presented here is clearly a variant of model-based learning. This enables the application of planning to reinforcement learning by referencing a known model of the system's dynamics (Botvinick, & Weinstein, 2014). Typically, these are quantitative models defined in terms of differential equations or neural networks. Our approach is definitely model based, because it learns the (quantitative) effects of taking particular actions in particular states, but it differs in the nature of the model, which is qualitative influences.

Historically, continuous state reinforcement learning has been addressed through function approximation, which requires careful selection of a basis function, such as., a CMAC (Santamaría, Sutton, & Ram, 1997) or radial basis function (Santos, 1999). In our work, qualitative models are used to discretize the continuous space. In fact, the entire point of qualitative models is to carve up a continuous space into semantically or causally meaningful phases. This has the benefit of being concise and communicable through language.

Although it is generally considered to be a form of unsupervised learning, Sutton and Barto (2018) take pains to claim that reinforcement learning is orthogonal to the supervised-unsupervised distinction. Instead, they argue that it is the exploration-exploitation tradeoff that largely defines reinforcement learning. Work in this area has focused on finding near optimal mechanisms for determining *when* to explore vs exploit learned knowledge (e.g., Kearns & Singh, 2002; Brafman & Tenenholtz, 2002). Our focus is instead on knowledge-directed experimentation that determines *what* to explore. In other words, most reinforcement learners treat exploration as randomly selecting from among actions or states that have not been encountered before. In our approach, exploration amounts to experimental design that strategically extrapolates or interpolates quantity conditions that rule an action in or out. It pursues reified learning goals and varies experimental conditions to minimize random exploration.

The nature of feedback in our system is more punitive than reinforcing. This is because the qualitative model implicitly provides initial rewards in the form of the goal hierarchy. There is no need to search blindly for actions that might positively influence the top-level goal. Instead, it learns to suppress actions when they are contraindicated by the particular quantitative state. Delayed effects in reinforcement learning are handled through a variety of methods, including eligibility traces (Singh & Sutton, 1996) and the method of temporal differences (Sutton, 1988). Our approach to handling such effects is the model-based credit assignment mechanism described in Section 4. This shares some aspects of model-free mechanisms, but benefits from the prior knowledge of the dynamics of the game.

Reinforcement learning typically requires hundreds to thousands of trials to learn even simple behaviors because it exhaustively explores the state space of the system. Hierarchical reinforcement learning is one technique for mitigating scaling problems due to high dimensionality by exploiting temporal abstraction and (Barto & Mahadevan, 2003). It would be interesting to see if the goal network in HRM could support the sort of stratified control found in such hierarchical systems. Another way reinforcement learning deals with the explosion of states is through function approximation, which is often used when actions or states can take on continuous values. Our use of qualitative states to encode action policies could be loosely thought of as a kind of knowledge-derived function approximation. This is a necessity for the HRM domain, which has 37 quantity fluent types, nine of them company wide and 29 being employee parameters. Given seven possible employees, this totals 212 concrete continuous quantity fluents, most of which can take on values from 0 to 100. Naïvely encoding states by partitioning the continuous quantities into buckets would be prohibitive.

AlphaGo and its successor AlphaGoZero are well known as highly successful reinforcement learners (Silver et al., 2016; Silver et al., 2017). They achieved superhuman performance in playing Go after millions of trials of self play, starting from what its developers claimed to be a blank slate. While extremely impressive, this is almost the exact opposite of our objective. Our approach purposely exploits prior knowledge in order to achieve acceptable performance on a game with very few trials. We believe that prior knowledge is not something to exorcise from a learner, but that it plays an important role in efficient learning. Marcus (2018) goes farther in arguing for the

importance of *innate* knowledge in his analysis of AlphaZero and the game knowledge that is inevitably built in to the learner (Marcus, 2018)

More generally, the issue of data efficiency has been addressed in other forms of learning. Transfer seeks to accelerate learning in one task or domain by reusing learned knowledge (such as facts, biases, features, or sub-plans) acquired from another task or domain (Pan & Yang, 2009; Konidaris, Scheidwasser & Barto, 2012). One-shot learning, (e.g., Li, Fergus & Perona, 2006) primarily focuses on object recognition and achieves data efficiency through expertise transferred from prior learning on other categories, reducing the incremental cost of learning to recognize a new category. Zero-shot learning seeks to master a task (typically recognition) with no prior training examples of the target (Xian, Schiele & Akata, 2017), by leveraging prior learning on other objects and features extracted from word embeddings. As with one-shot learning, the aim is to maximize data efficiency. The mechanism we present is (a) not learning a classification task, and (b) not transferring from a different task, but rather builds on a symbolic representation of a qualitative model. To the extent that the model could be acquired by learning from demonstration or instruction, it could be considered a type of transfer. Model acquisition is beyond the scope of this paper, but see Hinrichs and Forbus (2012) for an example.

Behavioral cloning is a method for learning by imitation that is often applied to learning to control a dynamic system (Mitchie, 1993; Bratko & Suc, 2003; Torabi, Warnell, & Stone, 2018). Our approach of using a qualitative model to guide learning and carve up a continuous space is especially similar to that of Bratko & Suc (2003), although HRM learns through experimentation, rather than imitation.

Our approach to the credit assignment problem bears some similarity to Langley's strategy learning program, SAGE (Sleeman, Langley & Mitchell, 1982). SAGE refined the conditions for applying a legal operator by comparing propositional differences between successful and unsuccessful applications. A key difference is that our use of a qualitative domain model guides credit assignment by making causal relationships explicit, and it lets the learning agent trace back in time to earlier states by following direct influences.

## 7. Conclusions

A qualitative model is one kind of prior knowledge that can guide learning. It is itself a form of declarative, acquirable knowledge that aid induction or support performance. One role is to facilitate experimentation, which reduces ambiguity in credit assignment by imposing controls on what to vary systematically and what to hold constant. We have presented two ways to achieve this: by generalizing or specializing qualitative state conditions on action selection and by manipulating the tradeoff ratios of activations for competing goals. In both cases, the result is to operationalize the qualitative model by learning more quantitative policies for pursuing actions or goals.

A major property of the learning technique described here is that it is data efficient. The approach attains good performance in under ten trials (or epochs in reinforcement learning vernacular), by virtue of starting with a qualitative domain model and ruling out vast portions of the potential state space whenever an action fails to provide a performance benefit predicted by the model. It need not wait until the end of the game to receive an extrinsic reward, since the model and its derived goal network provide immediate feedback via an audit trail from any action through intermediate quantities to the top level goal. We believe that the resulting data efficiency is an

important property of any learning system that purports to behave in a manner remotely like humans.

We designed HRM to be a simple testbed for exploring experimentation. The next step is to apply the capabilities described here in Freeciv[2], a much harder strategy game. Our intent is that self-directed experimentation will let a Companion (Forbus, Klenk & Hinrichs, 2009) learn continuously, over weeks and months, with minimal human intervention. It remains to be seen whether such learning will scale, if it will run afoul of the utility problem, or otherwise become unstable. We believe that prior knowledge, afforded by an extensive knowledge base and qualitative domain models, will mitigate these problems, as will natural language advice provided by a human mentor (McFate, Forbus, & Hinrichs, 2014).

Another avenue for future work will be treating learning goals more like domain goals, with their own activations, measurable properties, explicit processes, and strategies. This may ultimately permit further improvement by revising the order of experiments and conditions for stopping. This would be a step toward developing software organisms that not only behave independently but also learn independently.

## Acknowledgements

This material is based upon work supported by the Air Force Office of Scientific Research under Award No. FA9550-16-1-0138.

## References

Abel, D., Arumugam, D., Lehnert, L. & Littman, M. (2018). State Abstractions for Lifelong Reinforcement Learning. *Proceedings of the Thirty-Fifth International Conference on Machine Learning* (pp. 10–19). Stockholm, Sweden.

Angluin, D. (1988). Queries and concept learning. *Machine Learning*, *2*, 319–342.

Barto, A. G., & Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, *13*, 41–77.

Botvinick, M., & Weinstein, A. (2014). Model-based hierarchical reinforcement learning and human action control. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *369*, 20130480.

Brafman, R. I., & Tennenholtz, M. (2002). R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, *3*, 213–231.

Bratko, I., & Suc, D. (2003). Learning qualitative models. *AI Magazine*, *24*, 107–107.

Feifer, R.G., & Hinrichs, T. R. (1992). Using stories to enhance and simplify computer simulations for teaching. *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society* (pp. 815–819) Bloomington, IN: Lawrence Erlbaum Associates.

Forbus, K.D. (2019). *Qualitative representations: How people reason and learn about the continuous world*. Cambridge, MA: MIT Press.

Forbus, K, Klenk, M., & Hinrichs, T. (2009). Companion cognitive systems: Design goals and lessons learned so far. *IEEE Intelligent Systems*, *24*, 36–46.

[2] http://freeciv.wikia.com

Genesereth, M.R. and Thielscher, M. (2014). *General game playing*. San Rafael, CA: Morgan & Claypool Publishers.

Gil, Y. (1994). Learning by experimentation: Incremental refinement of incomplete planning domains. *Proceedings of the Eleventh International Conference on Machine Learning* (pp. 87–95). New Brunswick, NJ: Morgan Kaufmann.

Hinrichs, T. and Forbus K. (2016). Qualitative models for strategic planning. *Advances in Cognitive Systems, 4*, 75–92.

Kaelbling L.P., Littman, M.L., & Moore, A.W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, *4*, 237–285.

Kearns, M., & Singh, S. (2002). Near-optimal reinforcement learning in polynomial time. *Machine Learning*, *49*, 209–232.

Konidaris, G., Kaelbling, L. P., & Lozano-Perez, T. (2018). From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research, 61*, 215–289.

Konidaris, G., Scheidwasser, I., & Barto, A. (2012). Transfer in reinforcement learning via shared features. *Journal of Machine Learning Research*, *13*, 1333–1371.

Li, F., Fergus, R., & Perona, P. (2006). One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*, 594–611.

Marcus, G. (2018). Innateness, AlphaZero, and artificial intelligence. arXiv preprint arXiv:1801.05667.

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* (pp. 373–379). Québec City, Québec, CA.

Mitchell, T.M., Utgoff, P.E., & Banerjii, R. (1980). Learning by experimentation: Acquiring and refining problem-solving heuristics. In *Machine learning: An artificial intelligence approach*. Los Altos, CA: Morgan Kaufmann.

Michie, D. (1993). Knowledge, Learning, and Machine Intelligence. In L. Sterling (Ed.), *Intelligent Systems*, 2–19. New York: Plenum.

Pan, S. J., & Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, *22*, 1345–1359.

Santamaría, J. C., Sutton, R. S., & Ram, A. (1997). Experiments with reinforcement learning in problems with continuous state and action spaces. *Adaptive behavior*, *6*, 163–217.

Santos, J. M. (1999). *Contribution to the study and the design of reinforcement functions*. Doctoral Dissertation, Universidad de Buenos Aires.

Settles, B. (2012). *Active learning*. San Rafael, CA: Morgan & Claypool Publishers.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., & Dieleman, S. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*, 484–489.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., & Chen, Y. (2017). Mastering the game of go without human knowledge. *Nature*, *550*, 354–359.

Singh, S. P., & Sutton, R. S. (1996). Reinforcement learning with replacing eligibility traces. *Machine learning*, *22*, 123–158.

Sleeman, D., Langley, P., & Mitchell, T. (1982). Learning from solution paths: An approach to the credit assignment problem. *AI Magazine, 3*, 48–52.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning, 3*, 9–44.

Sutton, R.S., & Barto, A.G. (2018). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Torabi, F., Warnell, G., & Stone, P. (2018). Behavioral cloning from observation. *Proceedings of the 27th International Joint Conference on Artificial Intelligence* (pp. 4950–4957). Stockholm, Sweden.

Wang, Z., Garrett, C. R., Kaelbling, L. P., & Lozano-Pérez, T. (2018). Active model learning and diverse action sampling for task and motion planning. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 4107–4114). Madrid, Spain: IEEE.

Xian, Y., Schiele, B., & Akata, Z. (2017). Zero-shot learning-the good, the bad and the ugly. *Proceedings of the Thirtieth IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3077–3086), Los Alamitos, CA: IEEE Computer Society.