Advances in Cognitive Systems 4 (2016) 75–92 Submitted 8/2015; published 6/2016

Qualitative Models for Strategic Planning

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

Abstract

A longstanding challenge is to make intelligent agents reason more abstractly and farther ahead; in other words, to think *strategically*. The difficulty is that traditional approaches to planning focus on the effects of discrete, ground actions. High-level abstract goals may not be amenable to projecting concrete future states, yet they provide important long-term guidance in selecting actions, allocating resources, and focusing attention. We are exploring the representation and exploitation of strategy as a *qualitative modeling* problem. Our key hypothesis is that the same reasoning mechanisms can be applied to quantities at all levels of abstraction and that strategies can be expressed with respect to a domain-neutral vocabulary of processes, states, and goal tradeoffs in a fashion that is learnable, expressible, and explainable. We illustrate these ideas in the domain of Freeciv, an open-source strategy game.

1. Introduction

In adversarial situations with incomplete information, it is generally not possible to plan in detail very far into the future. Yet people are perfectly capable of making long-term strategic plans with partial commitments and vague intentions, and they adapt to, and compensate for, a dynamically changing world. Moreover, their goals often interact and trade off in complex ways.

In order to support such abstract and long-term reasoning, we are exploring explicit representations of strategies as *qualitative models* rather than as plans per se. Intuitively, the idea is that, while plans must bottom out in primitive actions, qualitative models are represented as relations between quantities which may be arbitrarily abstract. In other words, abstract models are represented with the same vocabulary as concrete models. This uniformity lets us use many of the same reasoning mechanisms to reason about intent as we do to reason about the mechanics of the domain.

For the purposes of this paper, let us define a strategy as a policy for making decisions about relative goal priorities, decompositions, and scheduling, typically at an abstract level. A strategy, therefore, need not refer to a complete plan, but may be as simple as a preference over partial decisions. We draw examples and illustrations from the domain of strategy games, although we believe the approach is applicable more generally. The particular game we are working with is Freeciv,[1] an open source implementation of Sid Meier's Civilization II.

There are three reasons to represent strategies explicitly: First, naming things makes them more communicable, and therefore more readily teachable. We aim to build systems that can learn

[1] `http://freeciv.wikia.com`

from high-level advice as well as from direct experience, and a named or reified concept can be better acquired through language. Second, strategies can help explain a system's behavior and make it more understandable. It is easier to explain reasoning by following a discrete audit trail of explicit decisions than by, for example, analyzing a matrix of coefficients. Third, if the strategies are sufficiently general, they should be applicable across domains. To the extent that strategies can be represented in terms of relationships between quantities and goals, rather than with respect to domain-specific low-level actions, they should be broadly applicable. Although it may be premature to make claims about generality, the compositional nature of qualitative models makes it possible to localize and isolate the connections between strategies and any lower-level domain-specific tactical behaviors. Consequently, learning how a strategy applies in a situation is likely to be easier than learning the strategy itself.

In the next section, we present some background on Freeciv, Qualitative Process Theory, and the Companion cognitive architecture. Section 3 describes the roles of a qualitative model and goal tradeoffs in domain reasoning, especially game playing. Section 4 introduces our representations for goals, goal tradeoffs, and the goal network, while Section 5 describes the application of this representational machinery to goals themselves in order to encode strategic knowledge. Section 6 presents some experimental evidence that such strategies can positively impact performance. Section 7 presents related research and Section 8 concludes with a discussion of limitations and future work.

2. Background

A quick introduction to Freeciv, Qualitative Process Theory, and Companions will provide some necessary background. Freeciv is a computer-based strategy game in which players build civilizations in a simulated world in order to ultimately either conquer opposing players' civilizations or to be the first civilization to launch a starship. In practice, the game is often played against three to four simulated players, which is the configuration we use. The world is defined on a randomly generated map partitioned into (by default) 4,000 *tiles*. Players start with an initial endowment of five *units*, which are agents within the game that can move and take actions. There are many types of units that can perform different kinds of actions or move on different terrain (e.g., land units and sea units). Each player starts the game with five initial units: two workers, two settlers, and one explorer. Settlers are uniquely able to build cities, in which terrain can be worked to produce wealth and new units and infrastructure can be built. Many different kinds of decisions are made in the course of developing a civilization, such as decisions about technologies to research, diplomatic relations with other civilizations, and investment decisions regarding infrastructure, military preparedness, seafaring and exploration. The fact that new entities, such as units and cities, are created dynamically and that opposing units and actions are often not visible to the player make Freeciv challenging for traditional models of planning. The wide variety of decision types and long-term nature of the game make it an excellent platform for exploring issues in abstract strategic reasoning.

The representations presented in this paper build on Qualitative Process Theory (QPT) (Forbus, 1984). Informally, QPT represents qualitative states as sets of ordinal relationships between quantities and propositional relations between entities. Consider, for example, a phase diagram of water as a function of temperature and pressure. The different regions, corresponding to solid, liquid, and gas, are qualitative states. In QPT, all quantity changes can be attributed to active processes that cause increases or decreases via *direct influences* between quantities, notated as `i+`

or `i-`. Direct influences on a quantity combine linearly to form a derivative, the rate of change of the influenced quantity. *Indirect* influences are qualitative proportionalities between quantities. The weaker form of indirect influences are `qprop` and `qprop-`, which indicate monotonic contributions of individual influences in the specified direction. A stronger form of indirect influence is represented by `c+` and `c-`, which are both linear and additive. In other words, those types of influences sum together directly to determine the exact value. In QPT, influences exist within the scope of quantity-conditioned *processes* and *model fragments* (Falkenhainer & Forbus, 1991). Processes are a type of model fragment that can contain direct influences, whereas basic model fragments can only contain indirect influences. An important distinction between a qualitative process model and a constraint network is that the process model explicitly indicates the direction of causality. This directionality is essential for reasoning about actions that may transitively affect desired quantities downstream.

The current research brings together qualitative modeling with Freeciv game playing in the context of the Companion cognitive architecture (Forbus, Klenk, & Hinrichs, 2009). A Companion system consists of functionally-specific distributed agents, such as ones for language interaction, analogical retrieval, sketch interaction, and meta-level reasoning. Each agent is endowed with a knowledge base containing the Research Cyc ontology[2] augmented with our own representations for qualitative reasoning, planning, domain knowledge, and learned knowledge. All agents support reasoning through deductive backchaining, hierarchical task network planning (Ghallab, Nau & Traverso, 2004), execution of plan primitives, and the ability to connect reasoning predicates to external code (called *outsourcing*). We specialized an agent to play Freeciv by emulating the Freeciv client, exposing the game state through outsourced predicates, and implementing actions as planning primitives that send packets to the local game server. Thus, a simple turn-based planning and execution loop can support basic game play, provided significant amounts of domain knowledge is encoded in plans and declarative knowledge.

In prior work, we focused on the problem of acquiring much of this domain knowledge through learning by demonstration (Hinrichs & Forbus, 2012a). The central idea there was that given a set of quantity types and a human-mediated demonstration of the game, the principles of QPT provide an inductive bias to identify qualitative influences and the effects of primitive actions on quantities. The inputs to the model learner are precise numeric samples of quantity values and changes in value after an action. The learning process is roughly similar to that in Bacon (Langley et al., 1987) except that, instead of producing equations, it creates outputs in the form of type-level influence statements between quantity types. *Type-level* here refers to a second order-relation that implicitly captures a universal quantifier (Hinrichs & Forbus, 2012b). The models that are learned in this way are somewhat oversimplified because the influences are not in the scope of model fragments or processes, and therefore are not conditional on any qualitative state. Nevertheless, after manually vetting and elaborating them, these learned influences form the core of the qualitative model used to play the game. The current model of Freeciv contains 104 influence statements, of which 64 were learned by demonstration.

Even with this knowledge, the level of play is severely limited because Freeciv is an incomplete information game with stochastic actions, an overwhelmingly large number of possible ground facts to consider, and many alternative dimensions along which to evaluate and compare possible plans. The spur for the current research is to learn more of the domain knowledge and plans, to focus the agent's attention on relevant abstractions that mitigate the problems of projecting uncertain information into the future, and to enable more reactive, short-

[2] http://www.cyc.com/platform/researchcyc

term actions to be guided by long-term goals. The qualitative model of game dynamics plays a key role in letting the planner quickly find and compare actions that drive quantities in desired directions without projecting exact and complete future states. We describe this process more concretely in the next section.

3. Reasoning More Abstractly and Farther Ahead

The basic idea behind using a qualitative model in game playing is to propose actions that drive quantities in desired directions. For example, given a goal to increase the population of a civilization, the planner will search through the known influences on the population quantity to determine that increasing the number of cities and increasing their rates of growth will both increase population. Searching further reveals that the primitive action of founding a city increases the number of cities (which is a quantity representing the cardinality of a set), while the growth rate of cities is qualitatively influenced by their food surplus and the presence of a granary. By traversing the transitive closure of influences, it is possible to collect the alternative primitive actions that can address the goal. Of course, these actions may not be currently legal, in which case a simple form of regression planning searches for known plans that can achieve the preconditions. Finally, because multiple actions may be feasible, heuristics order them roughly by cost – the number of actions to be performed and distances to be traveled. It is primarily this choice that we wish to make more intelligently through abstract strategic reasoning.

We refer to the strategic inference as abstract because the input to reasoning is not limited to domain-level events and states, but may instead be relationships between goals, specifically types of *goal tradeoffs*. Furthermore, the consequences of these inferences are not so much domain-level actions as relative priorities of goals and domain-neutral action types that, for example, decompose goals along some specified dimension. This contrasts strongly with other models in which a strategy is defined as the conditions under which a particular domain-level action is

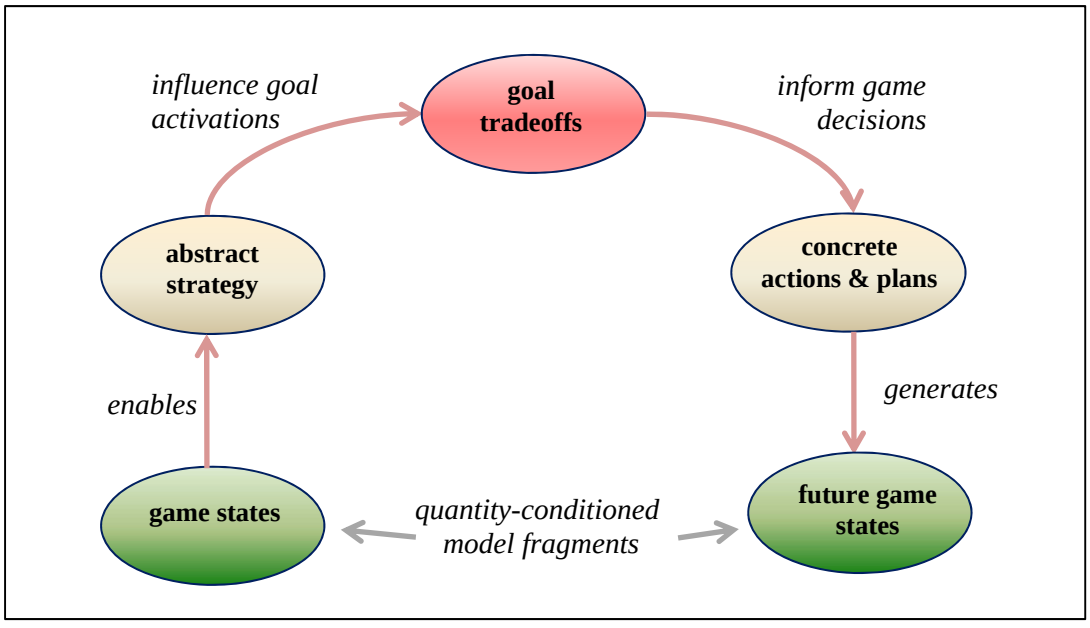

Figure 1: The role of strategies in generating behavior.

advisable (cf., Zhang & Thielscher, 2015). Part of our claim is that in order for strategies to transcend particular domains, they must be cast in terms of more general concepts, such as qualitative states and relations, goal decompositions, prioritization, and ordering. We do not expect an agent to invent new strategies from whole cloth (because this is a rare accomplishment in human experience) but, rather, to operationalize existing strategies with respect to a model or acquire them through interaction with an expert.

Figure 1 illustrates how strategies contribute to generating behavior in our behavioral model. The current situation in a game is summarized in terms of qualitative states defined by active model fragments and processes. The activation conditions in turn depend on ordinal relationships between quantities and Boolean relations between participant entities. For example, a civilization is in the *expansion phase* when it has fewer than ten cities. In general, a qualitative game state may involve multiple model fragments, expressed via type-level statements for conciseness. These states may in turn enable or activate strategies for resolving tradeoffs between goals. Strategies for resolving tradeoffs are themselves a kind of model fragment whose consequences either influence the relative priorities of the conflicting goals or further decompose a goal to subgoals that can be assigned different priorities. The goal priorities (or *activation levels*) inform decisions about the relative desirability of different actions, plans and assignments, thereby affecting the state of the game for the next round of decisions. For example, a high-level goal to grow a civilization will have competing subgoals to either increase the number of cities or their individual sizes. The appropriate choice shifts over time and this is reflected in the numeric activations of the two goals. While it may seem counter-intuitive to suddenly translate from qualitative states and relations to precise numeric activations, this is a necessary step whenever a model is used to drive behavior. Although analysis can be qualitative, behavior must ultimately resolve to concrete actions.

4. Goals, Goal Tradeoffs, and the Goal Network

In order to understand qualitative models of strategy, it is necessary to first explain the nature of type-level goals, how they trade off, how they are organized into a network, and the static analysis process that constructs these representations.

4.1 Types of Goals

Goals are partitioned into two mutually exclusive categories – *propositional goals* that describe states that can be achieved or avoided and *quantitative goals* that indicate the preferred direction to drive a quantity as shown in Table 1. Whereas a propositional goal can be directly planned for and satisfied, a quantitative goal is open ended and is therefore intrinsically a satisficing problem.

Table 1. Vocabulary of goal types.

(MaximizeFn (<quantity-type> <entity-type>))	Quantitative goal types
(MinimizeFn (<quantity-type> <entity-type>))	
(BalanceFn (<quantity-type> <entity-type>))	
(AchieveFn <proposition>)	Propositional goal types
(PreventFn <proposition>)	
(MaintainFn <proposition>)	

Propositional goals index plans and primitives for their pursuit or impose constraints on planned states, while quantitative goals enable the problem solver to retrieve processes and actions in the model that influence the goal quantity. In addition to these goal categories, we also classify goals by their *valence.* The valence of a goal is the direction in which a quantity is being driven or a proposition is pursued. So maximize and achieve goals have a positive valence, minimize and prevent are negative, and balance and maintain are neutral.

Like qualitative influences, goals themselves are represented at the type level. This means that they are implicitly quantified over descriptions of entities, rather than necessarily being specified for each individual entity. This is important both for reasons of scale (i.e., to avoid reifying distinct goals for each terrain tile, of which there are thousands in a typical game) and because the domain is dynamic – entities are created on the fly, and it is essential to be able to express goals involving them before they exist and have names. Table 2 characterizes the representation of a typical goal, in this case for maximizing the food surplus in the current player's cities. The statement associates a concise name (GoalFn 17) with an expression denoting a desire to maximize the values of instances of a quantity type (the food surplus produced by cities owned by the current player). The first argument to the MaximizeFn functor is the *quantity type* (MeasurableQuantityFn cityFoodSurplus), the functor of which designates this as a type of quantity that can be sampled via the second argument, the outsourced predicate cityFoodSurplus. The second argument to the MaximizeFn is the *entity specification*, which characterizes the objects possessing the quantity. In this case, GenericInstanceFn denotes a universal quantification over the elements of its argument, which is the subset of all cities owned by the current player. Rather than handcraft these somewhat verbose representations, the goals are constructed from seed goals and the qualitative model in a process of static analysis described in Section 4.4.

Table 2. Example goal representation for *"Maximize food surplus in your cities."*

```
(goalName (GoalFn 17)
 (MaximizeFn
  ((MeasurableQuantityFn cityFoodSurplus)
   (GenericInstanceFn
    (CollectionSubsetFn FreeCiv-City
      (TheSetOf ?var1
        (and (isa ?var1 FreeCiv-City)
             (ownsCity (IndexicalFn currentPlayer)?var1))))))))
```

4.2 Types of Goal Tradeoffs

Much of the skill in playing a game like Freeciv is knowing how to balance the conflicting demands of investing in future growth versus protecting a civilization from threats, keeping citizens happy, and researching new technologies. We classify these goal tradeoffs based on the kinds of resolution strategies they permit. Table 3 shows the types of tradeoffs we recognize, organized along two dimensions: *Total/Partial* determines whether or not all instances of the goal must be adjusted in lockstep, and *Abrupt/Progressive* determines whether the relative goal activation or prioritization happens instantaneously or progressively over time.

Table 3: Type-level goal tradeoffs determined by resolution methods.

Total Abrupt	**Partial Abrupt**
Switch relative goal priority from one goal type to another across all entity instances simultaneously	Change goal priorities for some goals or entities but not others
Total Progressive	**Partial Progressive**
Gradually change balance of goal priorities across all entity instances	Gradually adjust relative goal priorities across entities independently

For example, goals to maximize the percentage of revenue going to taxes, science, and luxuries trade off in a *total abrupt* fashion because the percentages must always sum up to 100 percent, and these rates are set at the level of a civilization, not independently for each city. Any change to one abruptly affects the others across the entire civilization. On the other hand, goals to achieve different improvements in cities only conflict locally within each city. So that indicates a *partial abrupt* tradeoff that could be resolved with a compromise, in which some cities might build granaries while others build libraries. A civilization-wide goal to prevent bankruptcy might become progressively more important as the treasury is depleted until it eventually overshadows other goals contending for money. The treasury is a global, divisible (but exhaustible) resource, which suggests a *total progressive* tradeoff. Finally, a goal to maximize the number of cities in a civilization trades off with goals to maximize the sizes, science outputs, or other properties of the individual cities, due to resource contention. Because the former goal is global, its priority could be progressively decreased. Because the latter goals are local to cities, they could be resolved independently, signifying a *partial-progressive* tradeoff. As with goal representation, the detection and labeling of pairwise goal tradeoffs is performed during static analysis, as described below.

4.3 The Goal Network

The type-level domain goals are organized into a directed acyclic graph of subgoal relationships, as shown in Figure 2. The goal network takes this form rather than a tree because any given quantity or achievable proposition may serve multiple goals. In addition to subgoal relations, the goal network contains explicit representations of pairwise tradeoffs between goals and provisional role assignments of equivalence classes of actor types (in Freeciv, players, cities or types of units) to goal types based on functional capabilities (see section 4.4 for details). The visualization of the graph shows quantity goals in green and propositional goals in blue, and it indicates goals that participate in tradeoffs with an oval shape. The figure is also annotated with topic clusters, which are illustrative but not an explicit part of the representation. Also, the network depicted here corresponds to a particular learned qualitative model. As additional technologies, infrastructure improvements and unit types are learned, they are reflected in new subgoal types.

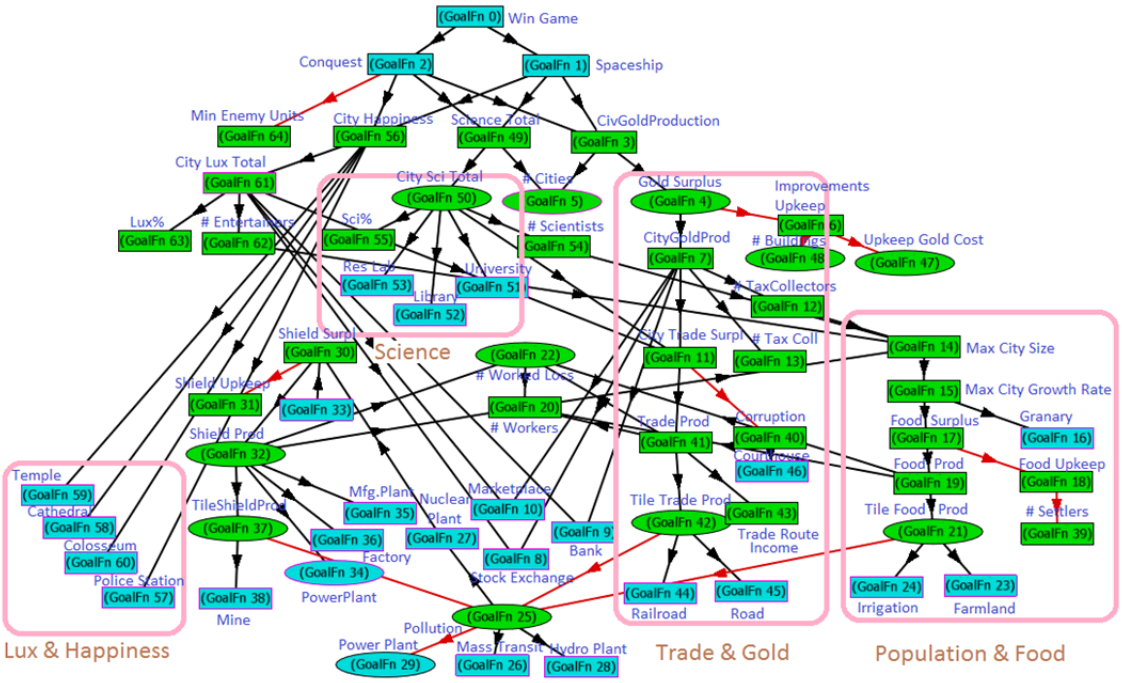

Figure 2: A Freeciv goal network, based in part on learned knowledge of the game.

Preventing loops in the subgoal relations requires reasoning about possible feedback loops in the qualitative model. In Freeciv, for example, artifacts can be built more quickly if a city has more surplus production capacity. Building a factory can result in greater surplus production capacity. Such a loop makes sense, but cannot be allowed to paralyze the reasoner. For this reason the routine that produces the goal graph detects feedback loops and labels the corresponding subgoal relations as *diachronic subgoals.* This is easy to detect because QPT requires that there be no pure indirect influence loops. Only a direct influence (i+ or i-) can close a loop, so these are turned into diachronic subgoals. Retaining these diachronic subgoals is important because it lets one of detect loops that serve no other purpose than to enhance the efficiency of attaining other goals. Activating such a loop constitutes a general *efficiency strategy* that can be recognized in expert behavior or applied to enhance future performance on other goals.

4.4 Static Analysis

The goal network is produced from the qualitative model and initial seed goals through a process of *static analysis.* At the start of a game, if there is no goal network for the domain, static analysis is automatically invoked to construct one. This process does three things: (1) it constructs the network of subgoals from seed goals and the qualitative model, (2) it identifies tradeoffs between goal types and labels them, and (3) it computes equivalence classes of actor types based on the capability constraints in preconditions of primitive actions, essentially creating a reverse index that maps from actor type to actions they can perform.

4.4.1 Generating the Goal Network

We have seen how the game player traverses the qualitative model to identify influencing quantities and possible actions. By definition, these influencing quantities and actions are *causally upstream* – i.e., they are independent variables with respect to the goal quantity and we can consider them to be subgoals. However, computing subgoals this way can be an expensive operation because the influences are specified at the type level where each influence specifies the relationship between instances of the two quantity types. For example, the influence of food production of a tile on the food production in a city is constrained by the requirement that the tile be inside the city boundaries. Consequently, computing subgoals directly from the influences requires reasoning at the instance level repeatedly for each entity instance at each edge in the graph. Static analysis avoids this problem by producing the context-free goal representation shown in Table 2. It does this by traversing the qualitative model *once* and accumulating the entity description from the relations in the individual influences.

The traversal must start somewhere, both to supply an initial goal valence and to specify an initial entity to which the influences are ultimately related. The latter is straightforward – we define an indexical referent to the current player. Traversing from the top-level goal is more problematic, because the goal of achieving 'winning the game' does not directly connect to the causal model. Freeciv has two main ways to win the game (conquest or escape to Alpha Centauri), so these become subgoals of the root goal. Conquest is represented as achieving the state where the cardinality of opposing players is zero, which entails the subgoal of minimizing the cardinality of opposing players. Players only exist when they have cities and/or units, a property known as *quantity-conditioned existence.* Consequently, the goal of minimizing opposing players expands to subgoals to minimize the cardinality of opposing units and cities. These subgoals *do* connect to the qualitative model by virtue of the fact that the attack primitive can reduce the cardinality of units and cities.

Nevertheless, the goal of attacking every opposing unit or city is insufficient to play an even marginally effective game, if for no other reason than that no opponents are visible at the start of the game and the initial units are not capable of attacking anyway. Consequently, we provide two or three initial *seed goals*, which can be thought of as waypoints. In our current experiments, these seed goals are to maximize the rate of gold production, maximize the rate of scientific research, and achieve the Republic style of government. Eventually, we intend to be able to specify such goal decompositions through advice in natural language, which would be a reasonable way to communicate strategy.

4.4.2 Detecting Tradeoffs

After the subgoal network is generated, static analysis proceeds to detect, classify, and label goal tradeoffs in the network. The recognition criteria we use are shown in Table 4, where tradeoff categories are abbreviated: TA (Total Abrupt), PA (Partial Abrupt), TP (Total Progressive), and PP (Partial Progressive). Type-level descriptions are notated with capital letters: G (goal), Q (quantity), P (proposition), E (entity), and |E| (cardinality of E). Lower-case e is an entity *instance*, while p is a *predicate*.

The simplest tradeoffs to recognize are opposite valences: goal pairs that both maximize and minimize the same quantity or achieve and prevent the same proposition. Others are more indirect, such as implied resource tradeoffs, where a unique property of an entity type indicates a resource bottleneck (e.g., a city's build queue is implied by the `currentlyBuilding` property which can only take on a single value at a time). Other tradeoff criteria are inferred by relation to

Table 4: Recognition criteria for goal tradeoff types.

$TA(G_1,G_2)$ =	$(oppositeValence(G_1,G_2) \wedge propositional(G_1) \wedge propositional(G_2)) \vee$ $resourceTradeoff(G_1,G_2)$
$PA(G_1,G_2)$ =	$(oppositeValence(G_1,G_2) \wedge quantitative(G_1) \wedge quantitative(G_2)) \vee$ $(partitionsQuantity(G_1,Q) \wedge partitionsQuantity(G_2,Q)) \vee$ $cardTradeoff(G_1,G_2) \vee achmntTradeoff(G_1,G_2)$
$PP(G_1,G_2)$ =	$maximize(G_1,\|E_1\|) \wedge achieve(G_2,P_2) \wedge actionInfluences(A_1,\|E_1\|) \wedge$ $resourceEnables(E_2,A_1) \wedge producesResource(A_2,E_3) \wedge (E_2 \subset E_3) \wedge$ $actionEffect(A_2, P_1) \wedge P_1 \supset P_2 \supset p(e) \wedge mutuallyExclusiveProperty(E_4,p) \wedge$ $e \in (E_3 - E_2)$
$oppositeValence(G_1,G_2)$ =	$(positiveValence(G_1) \wedge negativeValence(G_2)) \vee$ $(negativeValence(G_1) \wedge positiveValence(G_2)$
$sameValence(G_1,G_2)$ =	$(positiveValence(G_1) \vee positiveValence(G_2)) \vee$ $(negativeValence(G_1) \vee negativeValence(G_2)$
$positiveValence(G)$ =	$maximize(G,\_) \vee achieve(G,\_)$
$negativeValence(G)$ =	$minimize(G,\_) \vee prevent(G,\_)$
$resourceTradeoff(G_1,G_2)$ =	$mutuallyExclusiveProperty(E,p) \wedge$ $entailedGoal(G_1,E,p) \wedge$ $entailedGoal(G_2,E,p)$
$entailedGoal(G,E,p)$ =	$achieve(G,P) \wedge (e \in E) \wedge (P \supset p(e))$
$cardTradeoff(G_1,G_2)$ =	$cardinalityGoal(G_3,E) \wedge cardinalityGoal(G_4,E) \wedge$ $oppositeValence(G_3,G_4) \wedge$ $G_1 \in siblingGoals(G_3) \wedge G_2 \in siblingGoals(G_4) \wedge$ $G_1 \neq G_2 \wedge quantitative(G_1) \wedge quantitative(G_2) \wedge$ $\neg cardinalityGoal(G_1,\_) \wedge \neg cardinalityGoal(G_2,\_)$ $sameValence(G_1,G_3) \wedge sameValence(G_2,G_4)$
$achmntTradeoff(G_1,G_2)$ =	$oppositeValence(G_3,G_4) \wedge propositional(G_3) \wedge propositional(G_4)$ $subgoal(G_1,G_3) \wedge subgoal(G_2,G_4) \wedge G_1 \neq G_2$

existing tradeoff pairs, such as implied cardinality tradeoffs, where the quantitative siblings of conflicting cardinality goals are labeled as trading off (justified by the way cardinality goals originate in the qualitative model). We expect to extend these recognition criteria over time to support, for example, Total Progressive tradeoff detection.

4.4.3 Assigning Functional Roles to Goals

The third purpose of static analysis is to compute equivalence classes of actor types based on their functional capabilities. This process iterates through the preconditions of domain planning primitives and identifies constraints on the actor types that can perform them, merging them into equivalence classes. Then, when constructing the goal network, it identifies goal quantities influenced by actions and goal propositions achievable by actions and labels those goals with the functional capability role of actor types that can pursue them. This makes it efficient to quickly assign actors to goals they can pursue, so that planning can be resource driven from the available actors, yet still informed by relative priorities of long-term goals.

5. Representing Strategy Qualitatively

Strategies may be active or inactive based on states of the game or on quantitative relations between goal priorities. Thus the state of the agents' intent can be as relevant as the state of the game. In turn, the effect of a strategy is primarily to influence the priorities of goals. The representation of a strategy consists of a set of participants, relational constraints among them, a set of quantitative conditions for it to be active, and a set of consequences that influence or manipulate the participants. Table 5 shows a type-level representation of the Build Grow strategy, which is useful in the early expansion phase of the game. It defines participant roles, which are like pseudo-variables that can be bound to instances from the current game scenario. These roles, such as *doneBy* or *initialGoal*, are substituted with participant instances when querying for the activation status of the strategy and applying its consequences. An actual instance of a strategy or other model fragment can be specified using a positional notation, where the roles are assigned a unique ordering (shown here in the `associatedRoleList` statement). We see an example of positional reference in the second participant constraint, which says that the strategy is only applicable when the player's civilization is in the expansion phase. This is a bit of Freeciv knowledge in an otherwise domain-neutral strategy. Because this is an independent fact in a compositional representation, one could think of that as an applicability condition for the strategy in this domain.

The Build Grow strategy resolves partial-progressive tradeoffs between goals to maximize the number of instances of some entity type and goals to maximize properties of those instances. It emphasizes building entities before focusing on their properties. In the Freeciv domain, this is applied to building cities before expending resources to grow their population. While the strategy is active, its consequences influence the relative priority of the initial goal of the tradeoff to be 100 percent, and the deferred goal(s) to be 0 percent via the c+ (additive) influence type. When the strategy is deactivated, its influence is removed. When deactivating any strategy on a partial-progressive tradeoff, the initial goal is further suppressed with a new SuppressGoal strategy applied to it.

Table 5. The Build Grow strategy encoded as a qualitative model.

```
(isa BuildGrow StrategyType)
;; roles and types of participants:
(participantType BuildGrow doneBy Player)
(participantType BuildGrow initialGoal Goal)
(participantType BuildGrow deferredGoal Goal)
(associatedRoleList BuildGrow (TheList doneBy initialGoal))
;; relations between participants:
(participantConstraint BuildGrow
  (goalTradeoff initialGoal deferredGoal PartialProgressiveTradeoff))
(participantConstraint BuildGrow
  (activeMF (MFInstanceFn ExpansionPhase (TheList doneBy))))
;; influences on goal activation:
(consequenceOf-TypeType BuildGrow
  (c+ ((MeasurableQuantityFn goalActivation) initialGoal)
      (Percent 100)))
(consequenceOf-TypeType BuildGrow
  (c+ ((MeasurableQuantityFn goalActivation) deferredGoal)
      (Percent 0)))
```

Build Grow illustrates the similarity between strategies and QP representations. It is easy to see how the additive influences could be replaced with direct influences that would progressively drive goal priorities up or down over time. Other types of strategies, however, have consequences that are discrete actions rather than influences. The two action types that we foresee are goal *decompositions*, which are currently implemented, and the imposition of *scheduling constraints*, which are not. For example, a consequence consisting of (decompose <participant> <dimension>) creates a scenario-specific decomposition of a goal type along the specified dimension. This is used in the SpecializeByPerformance strategy to decompose a goal type into a subgoal containing just the entities that perform exceptionally well on that goal (see Table 6). The goal priority is emphasized for that subset of high-performing entities. For example, in Freeciv, the food, industrial production, and trade outputs of a city all trade off. Applying a SpecializeByPerformance strategy on the production goal will cause some cities to specialize in production rather than trade, and they will be more likely to build factories than granaries or markets.

Table 6. SpecializeByPerformance decomposes a quantity goal by performance outliers.

```
(isa SpecializeByPerformance StrategyType)
(participantType SpecializeByPerformance performanceGoal Goal)
(consequenceOf-TypeType SpecializeByPerformance
  (decompose performanceGoal DecomposeByPerformance))
```

As mentioned earlier, in order to drive behavior, it is necessary to resolve qualitative states into concrete numbers at some point. In the case of goal priorities, this is accomplished by representing them as percentages initialized to 100%, and then proportionally subdividing those percentages based on the different types of tradeoffs. Thus, if a goal does not trade off with anything, then it should be pursued no matter what. If a goal or its parent goals trade off with other goals, its relative priority must reflect that. The goal network is traversed breadth first and, as goal tradeoffs are encountered, the priorities are divided proportionally and multiplied by the maximum parent goal priority.

This provides a static picture of relative priorities, but because each game evolves differently, actual goals activations must change over time, based on the current situation and strategic intent. For example, if your civilization is attacked, then suddenly the importance of defense increases. Strategies account for this in two ways: by maintaining a separate record of the current dynamic goal activation in parallel with the static priorities and by elaborating the goal network structure with situation-specific decompositions of goal types. The dynamic goal activation differs from the static relative priority because it is sensitive to the existence or nonexistence of entities corresponding to goal types. For example, if there are no cities yet in the current game, then goals pertaining to cities should not impede unit or civilization-wide goals that can be pursued currently. The implications of distinguishing desired future states from future desires in this way are currently an area of active investigation. The other way goal activations evolve is through structural goal decompositions driven by strategies that are, in turn, activated by the current qualitative game state. Strategies for partial tradeoffs may decompose type-level goals first in terms of subsets of their entities and then allocate activation among their subgoals non-

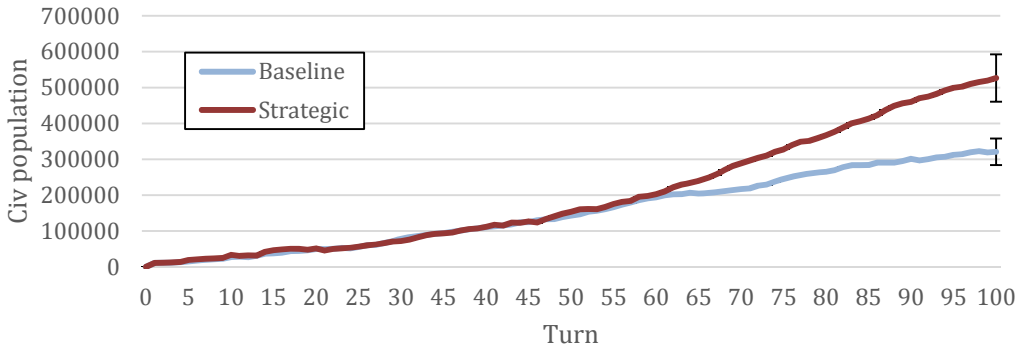

Figure 3. Population growth with and without strategies for resolving tradeoffs, averaged over 50 runs for each condition.

proportionally (i.e., by allocating activation to one subgoal type exclusively for that decomposition). Examples would include specializing some cities to production or trade and grandfathering existing Settler units to continue founding new cities, even after the parent goal deactivates the production of new Settlers.

6. Experiments and Analysis

As a step towards eventually learning strategies through advice, we wanted to establish that relatively domain-neutral strategies can in improve performance and are worth learning. To that end, we ran an experiment to compare performance in Freeciv with and without three hand-coded strategies. For the purpose of these experiments, we seeded the goal network with two high-level subgoals of the root goal to win the game: maximize the science output and maximize gold production. We had a Companion play 100 turns in 50 different scenarios (initial maps and unit placements). The baseline condition ran with the full complement of goals but no strategies for resolving tradeoffs among them, and the test condition ran with three strategies, directly associated with the goal tradeoffs they addressed. The strategies were BuildGrow, which addressed the tradeoff between founding new cities and improving them, ConditionOnState, which suppressed maximizing gold production unless the civilization was in the low-funds state, and the SpecializeByPerformance strategy, which decomposed goals to maximize production so that cities that were performance outliers with respect to that goal would prioritize it.

The results are shown in Figures 3 to 5. For each of these metrics, the performance difference on turn 100 is statistically significant ($p < 0.005$, $p < 0.001$, and $p < 0.001$, respectively). We can see that, on average, population and science output grows more quickly with the BuildGrow strategy in place, while gold production lags. This is to be expected: the treasury grows slower because more libraries are produced that require upkeep, effectively converting money to research. In the meantime, the ConditionOnState strategy suppressed the goal to maximize gold production unless the civilization was in the LowFunds state, defined as the treasury dropping below ten GoldPoints. In lieu of producing coinage, the Strategic tradeoff player built libraries to

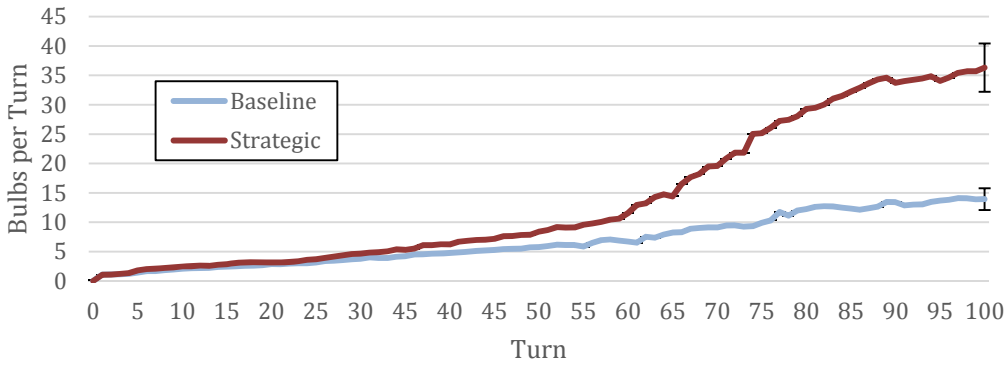

Figure 4. Science output rate with and without strategies for resolving tradeoffs, averaged over 50 runs for each condition.

boost science production. This is a desirable strategy in Freeciv – invest early in order to establish a technological lead.

Could these results have been achieved through more conventional means, such as pure hierarchical task network planning? Undoubtedly so, but not without doing severe violence to their underlying abstractions and at the cost of a target representation that would, for all practical purposes, be unlearnable. Consider that HTN planning is computationally equivalent to applying a set of situation-action rules that bottom-out in sequences of primitive actions. Almost anything could be implemented that way, yet it would be a poor choice for encoding domain-independent, abstract strategies. The problem is that HTN planners achieve abstraction at the cost of increased domain dependence in the form of high-level, domain-specific tasks and method expansions. Classical planners, on the other hand, achieve more domain independence by regressing over primitive actions. There the challenge is that actions must transform states in well-defined ways.

This tradeoff results from conflating two distinct notions: goal decomposition and operationalization. The traditional assumption has been that subgoaling reduces a goal from a possibly abstract desire to subgoals that are more operational or actionable, until the subproblems become trivial. This is not wrong, per se, but goal decomposition need not entail operationalization. In strategic decomposition, we split goals into subgoals not because the parent goals have no methods or actions, but because the subgoals allow for independent resolution of goal tradeoffs among entities at different points in time and space. This lets the agent explicitly reason about future goals, rather than automatically pursue desired future states. Thus future goals can be contingent on situations that might never arise and should not be pursued until they do.

Another way we distinguish decomposition from operationalization is by regressing through causal influences of a qualitative model. Different quantities may or may not be directly affected by primitive actions, but the subgoals that are pursued are not necessarily those that are most readily or directly manipulated. Instead, subgoal priority is determined by prior commitments to strategies that resolve goal tradeoffs according to more global, long-term, and abstract criteria.

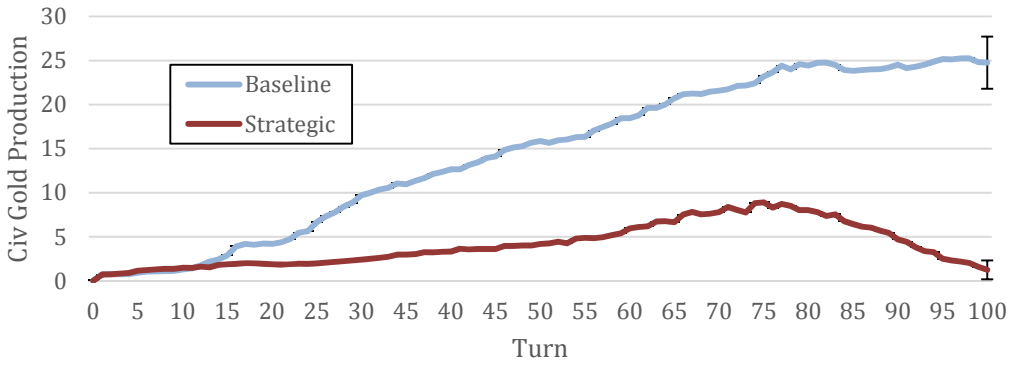

Figure 5. Gold production rate with and without strategies for resolving tradeoffs, averaged over 50 runs for each condition.

7. Related Work

Qualitative Process Theory (Forbus, 1984) provides the foundation of our modeling approach, including the basic vocabulary and constraints on representation. We have previously extended these ideas to cover modeling of adversarial tactics (Hinrichs et al., 2011) and to type-level influence and process representations (Hinrichs & Forbus, 2012b).

Strategic reasoning about goal relationships has a long history in AI. Notable early work included Wilensky's (1983) classifications of goal conflicts and meta-goals for resolving them, such as *achieve-as-many-goals-as-possible.* Our work extends those ideas to address tradeoffs between quantitative type-level goals and qualitative modeling of strategies.

Gruber's (1989) ASK program addressed the acquisition of strategic knowledge through user dialogs. Strategy in his model was more focused on search control rules than in generating behavior, but the acquisition process guides our approach to strategy learning. Similarly, Myers' (1996) Advice Representation Language enables the interactive communication of strategic advice to a hierarchical planner. The nature of advice in her language centers on constraining task expansion, but the way advice is mapped to problem solving method is relevant to operationalizing advice for resolving goal tradeoffs.

Static analysis has been explored previously, most notably in Etzioni's (1993) STATIC program, which applied explanation-based learning during static analysis to acquire search control knowledge. Like STATIC, our mechanism analyzes goal interactions, but the nature of the interactions and the purpose of the analysis are quite different. Rather than controlling combinatorial search, our analysis focuses on prioritizing goals for open-ended satisficing problems.

Goal-driven autonomy is an approach to generating and reasoning about goals in complex and dynamic environments (Klenk, Molineaux & Aha, 2013). One of the objectives is to relax the requirement that all goals be previously enumerated before execution. Our use of type-level goals addresses this problem by allowing entities to be created and destroyed and goals to be decomposed dynamically. Similarly, our planning system manipulates and reasons about goal

activation and prioritization. We differ in our focus on reasoning about goal tradeoffs and representing strategies as qualitative models.

GAIA is a system for the interactive model-based adaptation of software (Goel & Rugaber, 2015) that modified the Freeciv AI player code in response to changes in the game rules. To do this, it employed a declarative representation of strategy encoded in the Task-Method-Knowledge Language, reasoned about implications of the change to adapt the model, and generated new C code to implement the revision. Like GAIA, our system uses models to reason about high-level behavior, although it interprets them at runtime, rather than compiling new code. But while its models integrate teleology with structure and behavior, our approach maintains a purely causal qualitative model from which the goal network can be constructed. This has benefits and drawbacks. On the positive side, the separation of concerns makes it easier to learn the model from observation. On the negative side, plans we learn at the tactical level are pure hierarchical task networks (HTNs), which are difficult to modify and adapt because there is no model of their inputs, outputs, or purpose. Our application of Build Grow directly corresponds to two sequential steps in their "Largepox" strategy: Initial_M and Growth_M. Our purpose here is to try to tease out these constituents of strategy and represent them in the most general possible way.

A very different approach to learning in Freeciv was described by Branavan, Silver, and Barzilay (2012). That project aimed to learn the game through a combination of reading the manual and employing Monte-Carlo simulation. The most significant difference from the current work is that it progressively refined a heuristic evaluation function, rather than building or using any kind of model. Their system performed well, albeit on a scaled down game with a smaller map, that ended at 75 turns. A limitation of their approach is that it takes many trials to learn the game, requires running the game engine to do lookahead search, and, once learned, cannot explain its own behavior.

The GODEL system combines hierarchical and classical planning by using goal decomposition and planning landmarks to reduce the need for complete coverage of HTN methods in a domain (Shivashankar et al., 2013). Like GODEL, our approach combines HTN planning with classical regression planning by, among other things, decomposing goals and using waypoint goals in a manner similar to landmarks. Our approach differs in the nature of goals, goal tradeoffs, and explicit strategies. We use goal decomposition to achieve better tradeoffs, rather than to operationalize goals. Also like GODEL, we employ a partially ordered goal network, but the ordering (based on activation levels) is dynamically modified by strategies represented as qualitative models. These differences derive from different problems being addressed. Rather than planning extended sequences of actions, our game-playing system applies high-level strategic advice to constrain or influence lower-level behavior.

8. Conclusions

This paper has explored some potential benefits of representing strategies using qualitative models. Preliminary experiments show that this can positively impact performance in generating behavior. We have suggested that types of goal tradeoffs can constrain relevant strategies for resolving them, and that tradeoffs are often inferable directly from the structure of the qualitative model. Whenever goals involve satisficing relations on quantities, it is reasonable to expect that strategies will involve influences on quantities. The novel step here is treating the activation level or relative priority of goals themselves as quantities. This allows feedback from the external

world and the agent's own intentions to influence its goals, rather than solely constraining its means.

Of course, much work remains. For one thing, not all tradeoffs can be detected on the basis of a causal model alone. For example, the concept of *opportunity cost* is more a function of available plans and methods than causal relations, yet it can be central to goal tradeoffs. A clearer characterization of which tradeoffs can be detected directly from a model, and which require other information, is needed.

To cover a broader range of strategies, the representation of decompositions needs to itself be composable. It should be easy to express, for example, goals pertaining to military units in frontier cities in the late stage of the game. Reasoning about decompositions as first-class entities is critical to thinking abstractly and making partial decisions about plans. Strategies involving scheduling and synchronization are also underdeveloped. Although vocabularies for partially scheduling tasks are mature, it is less clear how to talk about future goals and tradeoffs between long-term investments and short-term gains. The solution will probably involve conditioning goals on future, envisioned states, and expressing temporal constraints between those states.

An important question is how many general strategies exist. The answer to that depends entirely on how abstract they are and how one counts. In fact, when treated as a representational design problem, the goal of having fewer primitive strategy types *trades off* with the goal of making them easier to express through advice. Parameterization and compositionality can result in fewer types at the cost of more complex participant bindings and referring expressions. The question that really matters is whether there is a sweet spot that facilitates the acquisition of strategic knowledge. Although the current research demonstrates the possibility of representing strategy qualitatively, finding the right combination of expressiveness and abstraction to compactly cover the space of general strategies remains an open problem.

Acknowledgements

This material is based upon work supported by the Air Force Office of Scientific Research under Award No. FA2386-10-1-4128.

References

Branavan, S. R. K., Silver, D., & Barzilay, R. (2012). Learning to win by reading manuals in a Monte-Carlo framework. *Journal of Artificial Intelligence Research*, *43*, 661–704.

Etzioni, O. (1993). Acquiring search-control knowledge via static analysis. *Artificial Intelligence, 62, 255*–301.

Falkenhainer, B., & Forbus, K. (1991). Compositional modeling: Finding the right model for the job. *Artificial Intelligence*, 51, 95–143.

Forbus, K. D. (1984). Qualitative process theory. *Artificial Intelligence*, *24*, 85–168.

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

Ghallab, M., Nau, D., & Traverso, P. (2004). *Automated planning: Theory and practice*. San Mateo, CA: Morgan Kaufmann.

Goel, A. K., & Rugaber, S. (2015). Interactive meta-reasoning: Towards a CAD-like environment for designing game-playing agents. In T. R. Besold, M. Schorlemmer, & A. Smaill (Eds.), *Computational creativity research: Towards creative machines*. Amsterdam: Atlantis Press.

Gruber, T. R. (1989). *The acquisition of strategic knowledge*. San Diego, CA: Academic Press.

Hinrichs, T. R., Forbus, K. D., de Kleer, J., Yoon, S., Jones, E., Hyland, R., & Wilson, J. (2011). Hybrid qualitative simulation of military operations. *Proceedings of the Twenty-Third Innovative Applications for Artificial Intelligence Conference* (pp. 1655–1661). San Francisco, CA: AAAI Press.

Hinrichs, T. R., & Forbus, K. D. (2012a). Learning qualitative models by demonstration. *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence* (pp. 207–213). Toronto, ON: AAAI Press.

Hinrichs, T. R., & Forbus, K. D. (2012b). Toward higher-order qualitative representations. *Proceedings of the Twenty-Sixth International Workshop on Qualitative Reasoning*. Los Angeles, CA.

Hinrichs, T. R., & Forbus, K. D. (2013). Beyond the rational player: Amortizing type-level goal hierarchies. *Advances in Cognitive Systems: Workshop on Goal Reasoning* (pp. 34–42). College Park, MD: University of Maryland.

Klenk, M., Molineaux, M., & Aha, D.W. (2013). Goal-driven autonomy for responding to unexpected events in strategy simulations. *Computational Intelligence*, *29*, 187–206.

Langley, P., Simon, H. A., Bradshaw, G. L., & Zytkow, J. M. (1987). *Scientific discovery: Computational explorations of the creative processes.* Cambridge, MA: MIT Press.

Myers, K. L. (1996). Strategic advice for hierarchical planners. *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning* (pp. 112–123). Cambridge, MA: KR, Inc.

Shivashankar, V., Alford, R., Kuter, U., & Nau, D. (2013). The GoDeL planning system: A more perfect union of domain-independent and hierarchical planning. *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence* (pp. 2380–2386). Beijing: AAAI Press.

Wilensky, R. (1983). *Planning and understanding: A computational approach to human reasoning*. Reading, MA: Addison Wesley.

Zhang, D., & Thielscher, M. (2015). A logic for reasoning about game strategies. *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence* (pp. 1671–1677). Austin, TX: AAAI Press.