NORTHWESTERN UNIVERSITY

Qualitative Spatiotemporal Episodic Memory

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

For the degree

DOCTOR OF PHILOSOPHY

Field of Computer Science

By

William Walker Hancock

EVANSTON, ILLINOIS

August, 2025

ABSTRACT

Qualitative Spatiotemporal Episodic Memory

William Walker Hancock

Episodic memory is crucial to an intelligent agent's ability to cope with a diverse array of tasks. It encodes knowledge of everyday experience and acts as a bridge between perceptual and conceptual worlds. Event representation and understanding is a key aspect of episodic memory, often involving continuous temporal and spatial aspects. In this work, I present a novel spatiotemporal event representation called Qualitative Spatiotemporal Episodic Memory, or QSTEM, that grounds continuous phenomena in local, qualitative, structured episodes. This representation is based on Hayes' notion of histories, or bounded pieces of spacetime, and is also inspired by psychological models of event perception, which describe how people carve up the continuous world into discrete events. Many existing statistical learners are black boxes which result in uninspectable models and require vast amounts of training data. By contrast, I show that QSTEM facilitates data-efficient analogical learning in two domains: a complex turn-based strategy game and a simulated cognitive robotics environment, resulting in inspectable models. The development of these representations addresses three main questions: 1. How are events individuated? 2. How should change within events be represented? 3. How can an agent learn from these representations?

ACKNOWLEDGEMENTS

To everyone along the way that has inspired and guided me. To my advisor Ken Forbus who has been unreasonably patient and has given me freedom to explore and tools to succeed. To my wonderful committee: Anthony Cohn, Ian Horswill, and Shiwali Mohan, who have been crucial for guiding this thesis to its final state. Thanks also to Charles Ortiz for early comments and suggestions for the thesis prospectus. To all of the members of QRG that I have interacted with over the years; I can say without hesitation that every single member of the lab has been a pleasure to know.

To my family and friends, most of all to my wife Steph who has helped and sacrificed in so many ways to see this through. Steph, Esben, and Elka have also given the necessary inspiration and motivation to get past hurdles and bumps in the road. Thanks to my parents, who over the years have provided the structure and freedom for a life well lived. To Stan and Sheila who have selflessly given their time, energy, and encouragement day in and day out.

And finally to our sponsor; this research was sponsored by the US Air Force Office of Scientific Research under award number FA95550-20-1- 0091.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

INTRODUCTION

Episodic memory is a persistent contextualized store of specific events (Tulving, 1983), often involving both spatial and temporal information. It is a powerful cognitive mechanism since it supports learning by experience. In qualitative reasoning research, Hayes proposed the notion of histories (Hayes, 1978;1989;1995) as a general framework for representing change in continuous domains. Whereas the situation calculus describes changes in terms of discrete events and provides no spatial constraints (leading to the infamous Frame Problem), histories represent change in terms of pieces of space-time, based on the objects involved in the changes. This makes histories a natural basis to consider for accounts of episodic memory.

Cognitive systems research has focused on representations of episodic memory that make minimal ontological commitments (Laird & Derbinsky, 2009; Menager & Choi, 2016), or are task specific (Brom et al., 2007). By contrast, I propose a novel history-based representation (Qualitative Spatiotemporal Episodic Memory, or QSTEM) that makes ontological commitments to descriptions of space and time while remaining task independent. QSTEM achieves this by utilizing the notion of histories to ground episodic memory in spatiotemporally local episodes. The idea is to use the spatial aspects of the entities involved in an event or situation and then describe how attributes of the event or situation change over time. This provides a representation of episodic memory that automatically segments time and space into discrete episodes and that also automatically adapts to change in spatial configuration.

Histories have been heavily used in qualitative reasoning (Forbus, 2019) and therefore should be valuable in representing episodic memories involving the continuous aspects of domains, including learning from experience. In this paper, QSTEM is used for analogical learning in two domains: a complex strategy game and a simulated cognitive robotics

environment. In the strategy game, the goal is to improve the agent's performance by leveraging QSTEM to learn how to play the game better. The idea is to automatically encode descriptions of bounded spatiotemporal regions so that the spatial and temporal extent of an episode varies in response to situations in the game. In the first experiment, the agent learns from a human, making the same kinds of decisions given a similar spatiotemporal context. In the second experiment, these representations are used to help the agent learn from its own gameplay experience. I show how QSTEM can be used to represent localized events, e.g. military battles, and how spatial perception affects how events are temporally segmented. In turn, this segmentation affects learning; representations that are too coarse or granular limit the ability of agents to generalize. In the cognitive robotics domain, the same QSTEM representations are used for concept learning. The robotic agent incrementally builds generalizations of events demonstrated by an instructor, like pushing a ball or spreading a group of objects on a table. The results show that these representations enable cross-domain data-efficient learning and result in inspectable models. In fact, event descriptions for both experiments use the same event structure, computation of compound entities, definition of persistence, and qualitative descriptions of change.

CONTRIBUTIONS

1. A domain-general qualitative spatiotemporal event representation called QSTEM
    a. Inspired by psychological models of event perception, it is
        i. Spatiotemporally bounded
        ii. Structured
        iii. Hierarchical, i.e. incorporating both a partonomy and taxonomy
2. A representation for a compound entity called a *footprint*

3. A discussion on and demonstration of the usefulness of qualitative spatial regions for event perception (i.e. temporal segmentation)
4. An algorithm for determining persistence for compound entities
5. An algorithm for learning limit points from episodic generalizations

CLAIMS

Claim	Experiments
1. Analogical learning with QSTEM is data efficient	1,2,3
2. QSTEM captures spatial aspects of events at a useful grain size	1,2
3. Important limit points occur on the temporal boundaries of events	2
4. Qualitative temporal descriptions of change facilitate learning	1
5. QSTEM facilitates learning	1,2,3
6. QSTEM facilitates human-like decision making	1
7. QSTEM aligns with intuitive notions of events such as military battles and spreading objects on a table	2,3
8. Qualitative spatial representations are useful for reasoning about event persistence, which in turn affects event segmentation and therefore learning	2
9. QSTEM aids decision making in a complex strategy game	1,2
10. QSTEM aids concept learning for robotics	3

The rest of this document is organized as follows.

Chapter 1: Research Background. Chapter one provides background on histories and the analogical learning processes used. It discusses the knowledge base used for describing background knowledge, and CogSketch, a tool for sketch understanding which provides spatial processing capabilities for the strategy game. Chapter one also introduces the two domains, a strategy game and a cognitive robot.

Chapter 2: Qualitative Spatiotemporal Episodic Memory. This chapter explains the overall event structure that is shared in both domains. It introduces qualitative spatial and temporal calculi, qualitative representations of change, a representation for compound entities, and algorithms for computing qualitative change from fluents, and definitions of persistence.

Chapter 3: Experiments in the Strategy Game Freeciv. This chapter describes two experiments in the strategy game Freeciv, where an agent uses QSTEM to improve decision making. In the first experiment, the agent learns from a human, making similar decisions as the human given similar spatiotemporal contexts. The second experiment shows how an agent can learn from its own gameplay, where it learns to improve its decision making by learning from military battle failures. The idea is that histories are well-suited for representing spatiotemporally localized events like battles.

Chapter 4: AILEEN: A cognitive robot. The second domain involves a cognitive robot called AILEEN. Here, the same QSTEM representations used in FreeCiv are used to teach AILEEN commonsense events like pushing a ball or spreading a group of blocks on a table.

Chapter 5: Related Work. This chapter situates QSTEM within three separate subcommunities in AI. The first is the qualitative reasoning community, which has previously used similar spatiotemporal event representations for a diverse set of tasks, including game AI and robotics as in this thesis. Second, the broad cognitive AI community has studied episodic

memory in tasks such as game playing, narrative understanding, and general machine learning. Third, the statistical machine learning community has studied event learning for self-driving vehicles and robotics, as well as general computational models of episodic memory.

Chapter 6: Conclusions and Future Work. This chapter discusses limitations with regards to computational models of episodic memory, and in learning limit points. I also discuss future work, with an emphasis on human AI collaboration.

CHAPTER 1: RESEARCH BACKGROUND

Research background is organized into four sections. The first introduces Hayes' notion of histories and then discusses how histories are useful for psychological models of event cognition. The second section describes the analogical reasoning stack that is used for learning in all experiments. Third, the two experimental domains are introduced: a strategy game and a cognitive robot. Finally, I discuss CogSketch (Forbus et al., 2011), a sketch understanding system which supports spatial representation for the strategy game, and Companions (Forbus and Hinrichs, 2006), a cognitive architecture which contains the FIRE reasoning engine and is built on Northwestern's NextKB knowledge base.

HISTORIES

The qualitative representation introduced in this work is based on Hayes' notion of histories, or bounded pieces of spacetime (Figure 1). Histories were developed around intuitive models of human thinking about space and time. They emerged in part as an ontological solution to difficulties in representing continuous entities. A canonical example involves liquids; pieces of lakes are not readily individuated, so it is more natural to reify them based on spatial context.

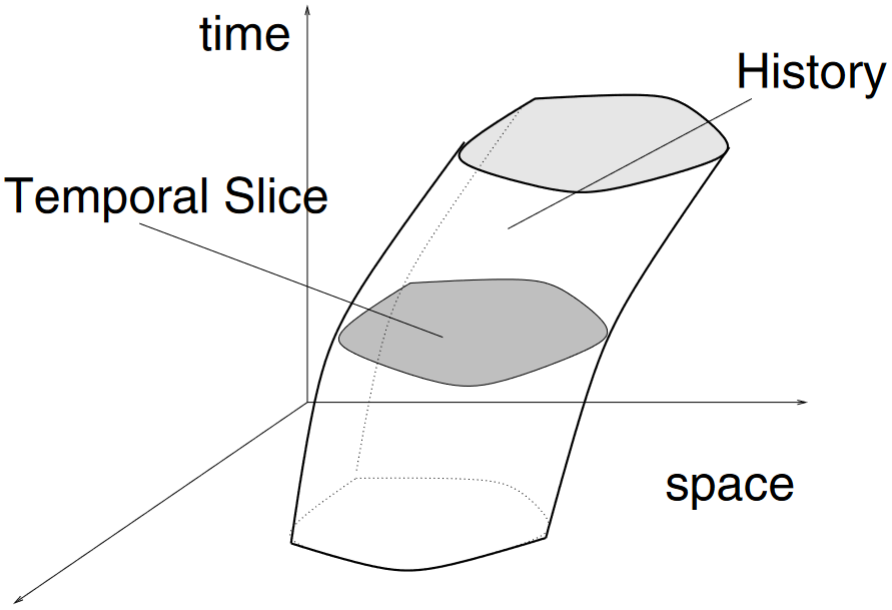


Figure 1: Depiction of a history for two-dimensional space (from Hazarika and Cohn, 2002).

Histories have been used extensively in qualitative reasoning research (Forbus, 2019) and so I hypothesize that they will be useful for learning with episodic memory.

From a commonsense perspective, many events can be represented using histories. There are often terms for spatial containers that accompany events: venues for parties, stadiums for sports events, battlefields for battles. All of these describe (often complex) phenomena that are constrained in space and have finite temporal extent. In the case of stadiums and sports, the

spatial bounds are unmoving and predetermined. For other types such as battles, there may not be an explicit container and the spatial component can change in size and grow. In fact, Hayes thought that there were likely many kinds of histories that aligned with commonsense notions of the world. For reasoning and learning, histories are useful as event representations for several reasons. First, spatial containment is a powerful constraint on the influences on processes. When hypothesizing about what causes an object to move, objects that never came into contact can often be ruled out[1]. This is a strong salve to the infamous frame problem and makes learning more efficient because it constrains case descriptions by filtering out irrelevant facts. When learning about the processes involved in a battle, what was happening elsewhere can be ignored (all else being equal). Bounded spatial representations also enable descriptions of localized change that adapt to changing configurations in the entities that they describe. For example, a military battle may start with a period where two opposing forces meet in battle, followed by one force surrounding the other.

In addition to constraining descriptions of what is salient in a context, histories allow for intuitive notions of continuity (Hazarika and Cohn, 2001). Describing the qualitative state of massing troops, i.e. an increase in cardinality over some time interval, requires specifying how compound entities persist. In other words, what makes a compound entity the same across time if it gains and or loses parts? In this work, I introduce a novel spatial representation for a compound entity, called a *footprint* which reifies compound entities. I then leverage qualitative spatial calculi to provide intuitive notions of persistence.

[1] There are notable exceptions that should be incorporated into full theories of naïve mechanics, e.g. gravity and magnetism.

QSTEM then provides a domain general mechanism for construing events as histories. The automatic spatial and temporal adaptation based on the entities involved means that these descriptions rely on general purpose encoding schemes, allowing agents to adapt to new situations more easily.

EVENT PERCEPTION

Event perception refers to people's ability carve up the continuous world into discrete episodes. For example, we speak of a day at the beach or a sixteenth birthday party. That is, in the continuous stream of time, we pick out individual events and generally know when they start and end and what happens during them. This ability begins early in life; infants discriminate between sequences of actions (jumps of a puppet) on the basis of numerosity (Wood and Spelke, 2005), suggesting that they are counting the distinct jump events. This is before the development of language, meaning that the participants haven't been taught the relevant boundaries verbally. Instead, it seems that there is some salient aspect of the event that is perceptible. One goal of this work is to show how qualitative reasoning supports this kind of understanding.

There is a rich body of research studying how people segment events, including Zacks' theory of Event Perception (Zacks, 2020; Swallow et al., 2009; Zacks and Tversky, 2001) and Event Segmentation Theory (Radvansky and Zacks, 2017). Event boundaries often correspond to changes in object movement, spatial location, characters, objects, causes, and goals. Sometimes these changes are discrete, e.g. a goal becomes satisfied, or a switch is turned on. In the latter case, we might segment time into a period where the switch is on or off. Segmentation for continuous quantities is more complicated, because not all change is relevant; representations should describe episodes at a useful grain size. There is evidence that people use qualitative representations of continuous quantities for common sense reasoning (Forbus, 2019) and I show

how they can be used for describing change, e.g. a period of decreasing height for a falling object. These kinds of qualitative representations (partitioning a quantity into periods of increasing/decreasing/quiescent change) describe what is happening (e.g. decreasing height) as well as segment time.

One goal of this thesis is to investigate the role of spatial reasoning for event cognition. Events and their participants are often spatially located and bounded. Location has been shown to be a powerful organizer of events in memory. Radvansky et al. (2017) showed that it is easier to remember the association between multiple objects at a single location than it is to remember multiple locations for a single object. For example, it is easier to remember that a plant, a poster, and an ATM are in a lobby than it is to remember that there is a plant in a lobby, a café, and a library. Radvansky suggests that differences in retrieval across the two conditions can be explained by event segmentation models. The idea is that the former case is organized using location (i.e. the library) into a single event, whereas the latter case is represented as three separate events. This emphasizes the primacy of space in our thinking about the world. In another study, Radvansky et al., (2011), introduced the *location updating* effect, where change in spatial location provides cues for segmentation. Items that had been placed in a backpack were more difficult to recall after experimental participants walked through doorways. Radvansky's explanation was that entering the room carves up time into two parts, before and after entering the room. Slower recall was explained as retrieval interference between the two distinct events. Here, once again, it seems that space influences organization of episodic memory.

The idea that space influences how people think about events was one of the inspirations for histories. One goal of this thesis is to give an account of how history-based representations can support models of event perception, i.e. how properties of space influence segmentation.

This often manifests as simple qualitative spatial change affecting segmentation, e.g. up/down movement, away/towards, in/out. However, one contribution of this thesis is an account of how histories support reasoning about segmentation at a more fundamental level. Indeed, one insight of Hayes' was that intuitive notions of persistence for continuous entities depend on space. Recall that continuous entities like bodies of water can be reified by their containers, e.g. a lakebed. This container then allows tracking the lake's existence over time, e.g. it persists as long as there is water within the lakebed. Then, persistence can be defined in terms of temporal continuity for spatial regions (Hazarika and Cohn, 2001). While the focus for histories was originally on continuous entities, histories can also describe groups of discrete objects by referencing their container. We talk of ant colonies, military units, flocks of birds, crowds of people, etc.[2] For these cases, a container is reified, and change is tracked just as in the continuous case. Reifying the container provides a useful means of stratification; people represent episodes at multiple grain sizes, i.e. the aggregate or the individual. Partonomic spatial relations between the container and individuals then provide a domain-general way to reify this relationship. These family of spatial calculi are introduced later in this chapter.

ANALOGICAL LEARNING

One goal of this research is to demonstrate how analogy can support spatiotemporal learning. To do this, I use the Structure Mapping Engine (SME) (Forbus et al., 2017) for comparison, the MAC/FAC retrieval system (Forbus et al., 1995), and the SAGE generalization system (Kandaswamy and Forbus, 2012). These analogical mechanisms are tightly integrated in

[2] There is a sense in which most entities are compound. Cats lose their tails, people lose hair, etc. In philosophical ontology, this arises in discussions on the distinction between events and objects. For naïve theories, the implication is that people likely use both constructions, and therefore that QSTEM is a useful construct for commonsense entities in general, not just flocks, groups, etc.

the underlying reasoning engine and provide the mechanisms for retrieval, comparison, and transfer. There is a growing body of evidence that human cognition and therefore commonsense reasoning depends on structured representations of the world (Forbus et al., 2017). The perceptual world around us is highly structured, both temporally and spatially, and so the hypothesis is that analogical processes play an important role in learning.

SME

The Structure Mapping Engine (SME) (Forbus et al., 2017) is a computational model of matching in Structure Mapping Theory (Gentner, 1983). SME provides a mechanism for determining similarity between two structured descriptions. It does this by generating a set of mappings between a base and target. A mapping aligns entities and expressions between the

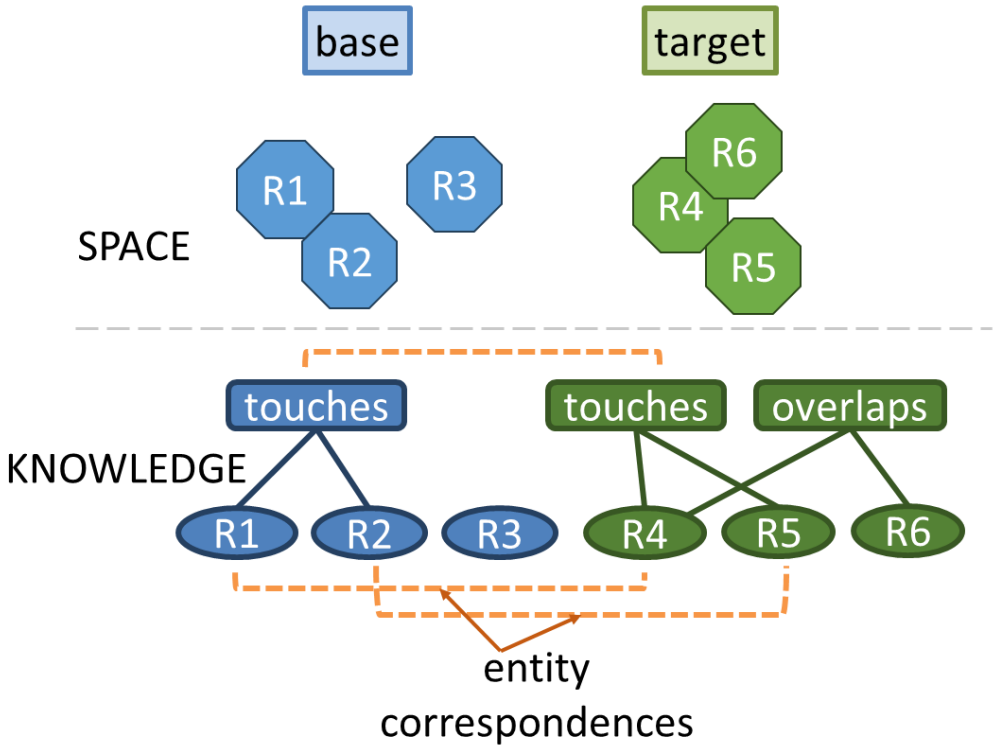


Figure 2: Example spatial configuration (top) and corresponding SME comparison between the base and target descriptions (bottom).

descriptions. A structural similarity score is computed for each mapping that prefers higher-order structure and entities that participate in systems of relations. SME additionally computes a set of candidate inferences, or surmises about the target made on the basis of common structure plus the base representation. Figure 2 shows an example mapping between base and target for two spatial descriptions, with entity and relational correspondences.

MAC/FAC

Analogical retrieval is performed by MAC/FAC (Figure 3) (Forbus et al., 1995), which stands for "Many are Called, Few are Chosen" because it uses two stages of map/reduce for scalability. The inputs consist of a probe case and a case library. The MAC stage, in parallel, computes dot products over vectors that are automatically constructed from structured

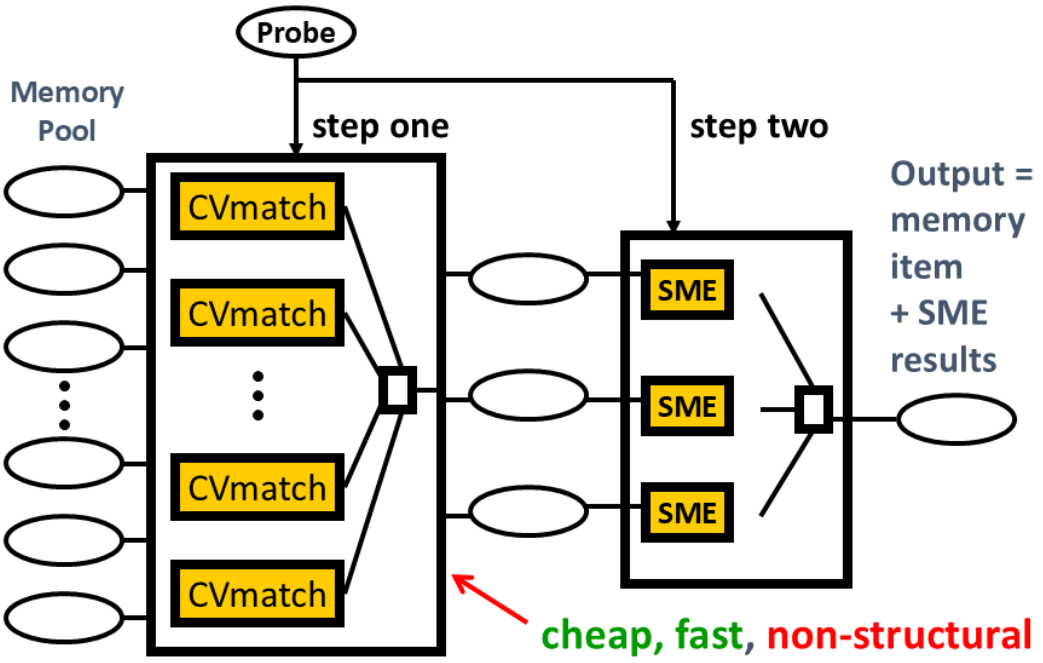


Figure 3: Two steps of the MAC/FAC algorithm. The first step uses vector-based similarity for fast comparison in order to scale; the top three most similar entities are then compared against the probe using SME, where the most similar is returned along with a similarity score.

descriptions, such that each predicate, attribute, and logical function are dimensions in the vector and whose magnitude in each dimension reflects their relative prevalence in the original structured description. The best mapping, and up to two others, are passed to the FAC stage. FAC compares the best structured descriptions from the MAC stage to the input probe using SME. The best match plus up to two others, is then returned. The inexpensive nature of the MAC stage allows for scalability, while the FAC stage allows for sensitivity to structural similarity.

SAGE

SAGE, or the Sequential Analogical Generalization Engine (Kandaswamy and Forbus, 2012) is an analogical generalizer. It utilizes MAC/FAC for retrieval, and therefore SME at its core. SAGE operates over generalization pools (aka gpools), each of which represents a concept. Each gpool consists of a set of generalizations and outlier examples. Given a new example and gpool, MAC/FAC first finds the most similar generalization or outlier to the example. This retrieval results in a SME mapping, which aids in assimilation. Examples are only merged if their similarity score exceeds an assimilation threshold. Generalizations accumulate statistics over expressions and filter out low frequency expressions based on a probability cutoff. Unlike many ML algorithms, SAGE is an incremental learner, which eliminates the need for retraining a model. It is similar to k-means with outliers, but unlike k-means uses a human-normed similarity metric operating on structured representations. In addition, the number of generalizations falls out automatically (i.e. it does not need to be prespecified), allowing the flexible learning of disjunctive concepts. As part of the generalization procedure, SAGE lifts mapped entities, and replaces them with generalized entities (GenEnts), a form of Skolem constant. GenEnts that correspond to quantitative arguments store statistics on the participating quantities. Specifically, cardinality, minimum, maximum, mean, and sum of squared error with respect to mean are

collected. A generalization can be queried so that it returns these statistics for a specific quantitative generalized entity.

FREECIV

I use Freeciv[3] as a testbed for the first two experiments. Freeciv is a complex turn-based strategy game based on the Civilization franchise. It involves playing as an emerging nation, vying against potentially hostile nations to either conquer the world or build a spaceship and found a new colony in space. Successfully achieving either outcome requires a player to manage their scientific progress, military, and economic growth all while conducting diplomatic relations. Figure 4 shows an early game scenario where the Swiss player has four cities in its empire.

Figure 4: An early-game Freeciv scenario where the player's civilizations has four cities.

[3] https://freeciv.org/ (visited on 4/1/2025)

Freeciv is an excellent domain for AI research due to its complexity. A typical game board consists of 4,000 tiles with varying terrain. Games typically last for hundreds of turns, and each turn involves many decisions. Some decisions are global across the entire civilization, such as setting the tax rate, determining the next technology to research, and engaging in diplomacy. Workers must be kept busy modifying terrain. Military units must defend cities and conduct attacks on opponents when at war. By contrast, Go is played on a 19x19 grid with uniform, immutable spatial properties which are always visible from the start. In addition, each turn in Go only involves a decision to place one piece.

Freeciv is especially useful for exploring histories because important behaviors happen at multiple grain sizes. For example, there is typically an expansion phase, where a player builds out multiple cities to stake out desirable territory and deny it to competitors. Wars can cause the expansion or contraction of a player's civilization, depending on their success, making decomposing time based on the set of cities a useful distinction. While the expansion phase requires reasoning at the level of an entire civilization, reasoning must also occur at the level of individual units and cities. Making decisions in the game should benefit from analyzing experiences built on histories because the resulting descriptions adapt to entities in the environment and their changing properties.

Recent work in real time strategy games (RTS) including OpenAI's Dota II player OpenAI Five (OpenAI et al., 2019) has shown impressive game performance, beating many human players. These models take immense computational resources to learn and the decisions that the agent makes are uninterpretable. The representations used in OpenAI Five are ad-hoc; they are on one hand global, in that they include information about every unit visible on the map, but they also make locality assumptions both in time and space, by representing what is in the

immediate vicinity of individual units. The computational expense of learning these models means that these locality assumptions are not tested, and the uninterpretable nature of neural models means that it is unclear whether they affect gameplay. Additionally, the vector-based representation used constrains the kind of structured knowledge that is available for learning. The 8x8 grid around each hero is included in the representation, but this region is fixed across the duration of the game and fixed for each hero type. Temporal locality is also fixed for certain attributes, e.g. each unit's health over the previous 16 timesteps is included. Again, it is uncertain how these assumptions affect the agent's ability to learn, or if they hinder decision making.

There is a rich history of applying AI for playing strategy games, for example reading the manual to improve gameplay performance (Branavan et al., 2012), responding to unexpected events (Klenk et al., 2013), using qualitative models to reason about decision making (Hinrichs and Forbus, 2016), metareasoning for model building (Goel and Jones, 2011; Ulam et al., 2008), and case-based reasoning for decision making (Ontañón and Ram, 2011; Ontañón et al., 2007). The breadth of these topics further supports the application of video games as AI testbeds. To my knowledge, the application of history-based event representations to decision making in video games is unexplored.

AILEEN

AILEEN (Mohan et al., 2020) is a cognitive system that learns new concepts through interactive experiences with a trainer. AILEEN lives in a simulated robotic world built in Webots[4] (Michel, 2004) and is an embodiment of the Soar Cognitive Architecture (Laird, 2019). Soar has two declarative long-term memories - episodic and semantic - each with its own

[4] https://cyberbotics.com/

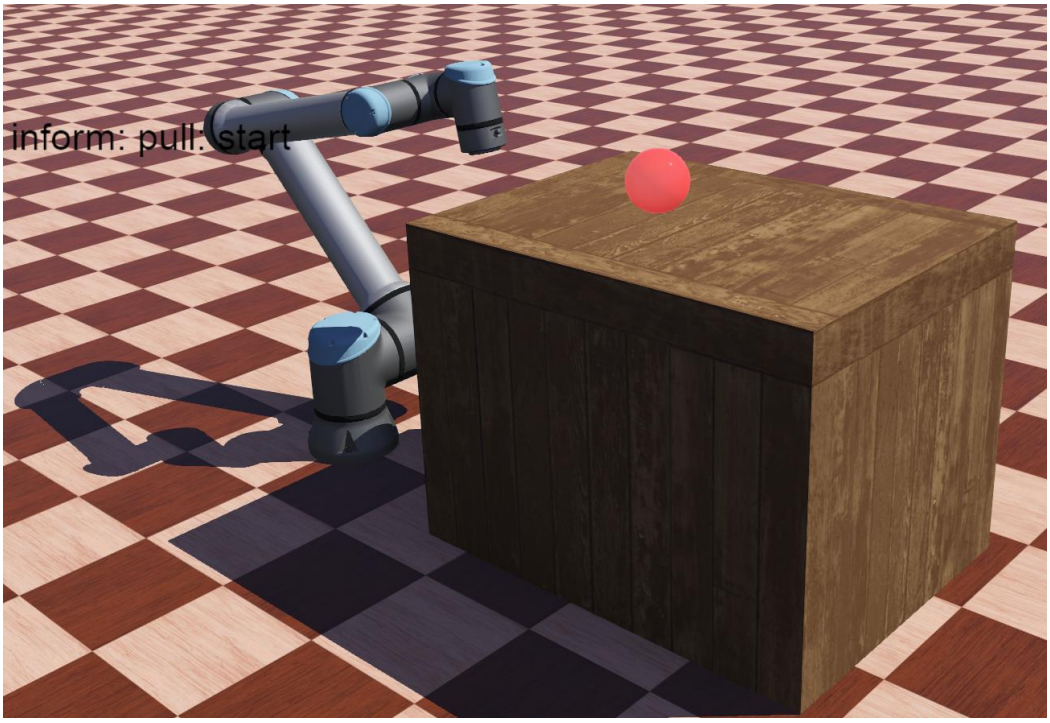


Figure 5: The Webots environment where interactive learning takes place. The instructor places objects on the table, and AILEEN records traces of the demonstrations for a concept learning task.

distinctive function. Episodic memory provides access to past contextual experiences. Semantic memory represents and provides access to general factual knowledge about the world. This work focuses on developing AILEEN's analogical episodic memory so that it can acquire general knowledge about the world by collecting and analyzing experience. It is related to ROSIE, a cognitive system that has demonstrated interactive, flexible learning on a variety of tasks (Mohan and Laird, 2014), and implements a similar organization of knowledge. AILEEN's episodic memory uses the same analogical mechanisms as in the Freeciv experiments - the Structure Mapping Engine (SME), MAC/FAC, and the Sequential Analogical Generalization Engine (SAGE).

Webots is an open-source 3D mobile robot simulator. It was originally created as a research tool to investigate control algorithms for mobile robotics. It has since become a

platform for a more diverse array of robotics research. As a simulator, Webots allows the quick creation of many different scenarios. There are pre-built environments for outdoors and indoors, and many different 3D models that can be incorporated into a scene. It also incorporates a physics simulator which allows rigid body mechanical processes. Webots is also extensible; its API allows easy integration of outside software, e.g. cognitive architectures. Figure 5 shows an example scene for an AILEEN instruction lesson.

NEXTKB

This work uses Northwestern's open-source NextKB knowledge base, which contains over 80,000 concepts and over 25,000 types of relations, with over 700,000 facts. It is built on OpenCyC, the freely distributed subset of the Cyc knowledge base (Lenat, 1995), and contains a wide variety of additional knowledge, e.g. facts supporting qualitative and analogical reasoning and learning and perceptual processing. Cyc is a long-standing effort aimed at capturing general human knowledge in a structured, machine-readable form. Cyc encodes a comprehensive ontology of commonsense knowledge; unlike more specific or domain-oriented knowledge bases, Cyc attempts to codify the broad range of facts and relationships that underlie everyday human cognition, including concepts like time, space, and causality, as well as social, psychological, and physical phenomena.

The Cyc KB is divided into microtheories, which represent specialized domains or contexts. Microtheories can be linked via inheritance relationships to form logical environments which support and control reasoning. Examples of environments used in this work are game scenarios and spatially bound events. Microtheories are also used to represent cases for analogical learning. Using representations partially derived from OpenCyc enables leveraging

the several person-centuries of work that has gone into its development and reduces the risk of tailorability, as does using domain-general qualitative representations (Forbus, 2019).

COGSKETCH

CogSketch (Forbus et al., 2011) is a sketch understanding system that provides a model of high-level visual processing. It provides multiple, hierarchical levels of visual representation, including decomposing digital ink into edges, combining edges into entities, and grouping based on gestalt principles. The basic level of organization for ink in CogSketch is the glyph, one or more ink strokes that are taken to represent some entity (abstract or concrete) in the sketch. Glyphs can be generated by people using a pen or mouse to produce digital ink. But glyphs can also be automatically produced via visual analysis of images (Chen et al., 2019). CogSketch is capable of computing various spatial relationships between glyphs, including adjacency, relative position, topological relationships and relative size. Properties of individual glyphs can also be computed such as shape attributes (roundness) as well as glyph decompositions such as shape skeletons and Voronoi diagrams. Moreover, CogSketch can decompose glyphs into edges, and group visual entities based on gestalt properties. These capabilities have enabled it to model a variety of visual problem-solving tasks, including Ravens' Progressive Matrices, one of the most common tests used to measure human fluid intelligence. The CogSketch model uses analogical reasoning at multiple levels of visual representations, including re-representing visual descriptions automatically as needed. Its performance places it in the 75th percentile for American adults, better than most adult Americans (Lovett & Forbus, 2017).

CogSketch interfaces with the Freeciv simulator via a Lisp-based API. A sketch, shown at the top Figure 6, contains layers for different kinds of objects in Freeciv, e.g. one layer each for units and cities. Spatial representations for this work build upon those developed by McLure

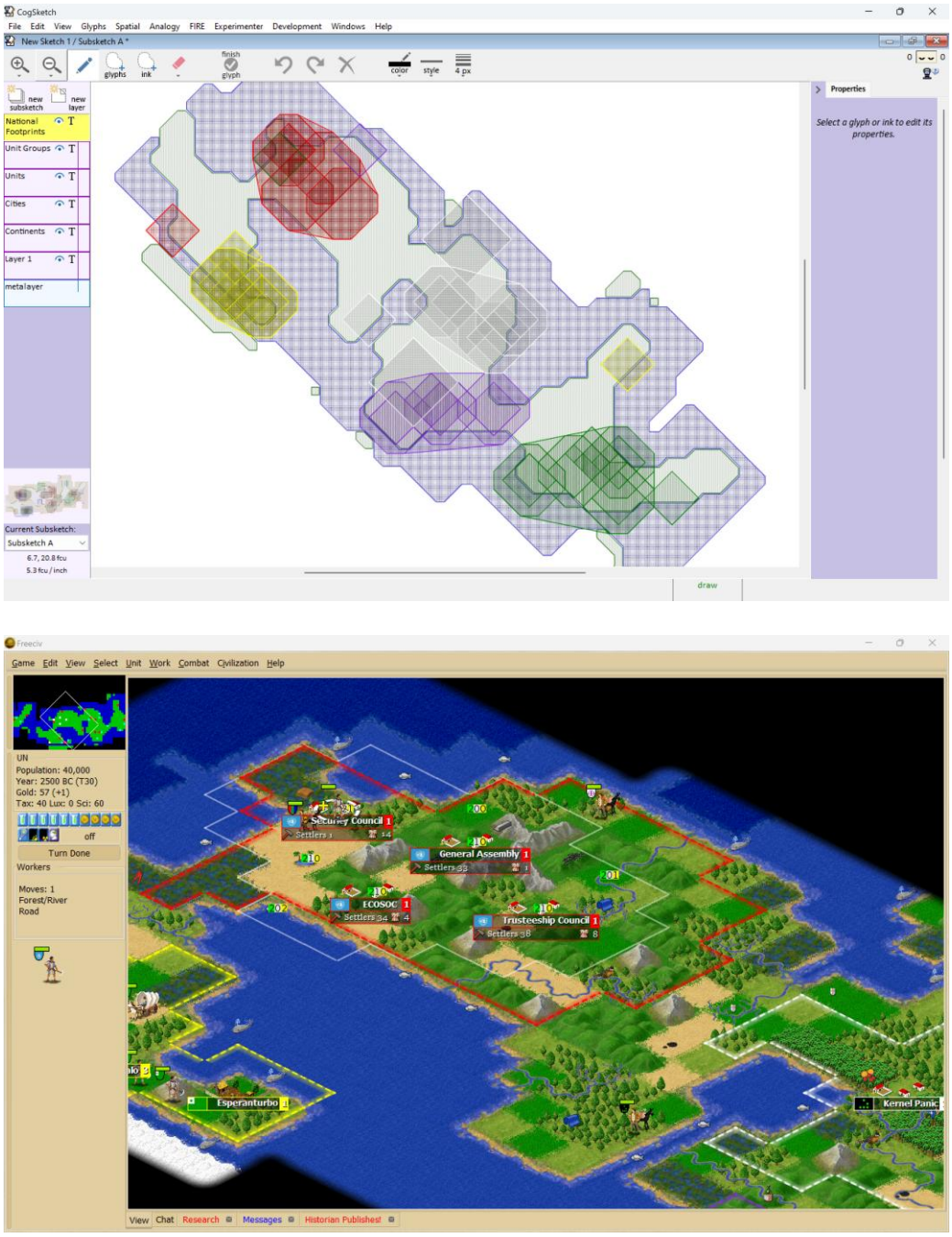


Figure 6: Below, a Freeciv Scenario and Above: a spatial representation of the scenario in CogSketch, which is automatically computed and updated in real time.

where geospatial terms like island and isthmus were learned from spatial representations of the map via analogical generalization (McLure & Forbus, 2012). Examples spatial representations were blobs for continents and oceans. QSTEM uses the same underlying abstract blob

representations to represent objects in the game like cities and the player's civilization as spatial regions, discussed next.

CHAPTER 2: SPATIOTEMPORAL EPISODIC MEMORY

This chapter introduces QSTEM, starting with how entities are represented as spatial regions and related using qualitative spatial calculi. Then, representations of how spatial entities change are introduced, e.g. a period where a ball is rolling on a table. The overall goal is to represent events as pieces of spacetime, and we start by considering space first. For generating QSTEM cases, there are three main considerations:

- How spatially extended is an episode?
- What descriptions of change are used?
- How temporally extended is an episode, i.e. how is segmentation computed?

The next section describes the spatial aspects of a QSTEM case, and the following section describes how change is represented and how segmentation is computed.

SPATIAL REPRESENTATIONS

The spatial component of a QSTEM episode consists of regions and relations between them. Regions can be primitive, e.g. tiles in Freeciv or physical object boundaries, as well as compound entities constructed from sets of primitive regions. For relations between them, the region connection family of calculi is used, discussed next.

SPATIAL CALCULI

The region connection calculi are mereotopological systems describing binary parthood relations between regions. There are many relation algebras that fall under this family of calculi.

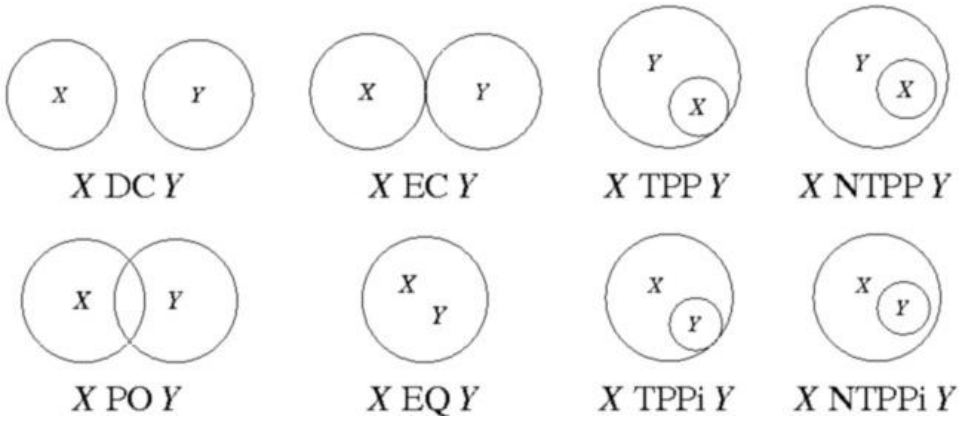


Figure 7: Region Connection Calculus with 8 relations. RCC8 describes binary relations between spatial regions.

For this work RCC8 (Cohn et al., 1997) is used (Figure 7), which allows encoding concepts like containment, overlap, disconnectedness, touching, etc., and RCC2 which simplifies RCC8 into two relations, RCC2-connected and RCC2-disconnected. The choice of spatial relations is driven by two main factors: one, intuitive notions of how people think about the world, and two, design considerations for good representations for learning.

First, regions and qualitative relations between them align with people's intuitive notions of space (Forbus, 2019). Qualitative calculi like the region connection family admit containment and allow for exploiting hierarchical structure of spatial relationships. In a battle, for example, any participant is spatially contained within the overall event, which in turn might be part of a larger war. The spatial components of events, however, are not purely hierarchical. Coincidence, both in time and space, must accommodate overlap, e.g. opposing military forces within a single battle, and RCC8 provides a mechanism for describing this kind of configuration, e.g. the partial overlap relation (rcc8-PO).

Second, for considerations of learning, there is a tradeoff between sparsity and conciseness. Sparsity promotes learning because sparse representations can more easily

distinguish differing cases. Jointly exhaustive and pairwise disjoint (JEPD) sets of relations such as the region connection calculi family ensure sparsity by their underlying axiomatic assumptions. Conversely, calculi that make more distinctions tend towards dissimilarity. The implication is that sparsity must be optimized. As calculi make more and more distinctions (e.g. RCC13 vs RCC8), cases become dissimilar, all else being equal. Here, the relevance principle (Forbus, 2019), which states that case descriptions should not make unnecessary distinctions, guides our selection of RCC8.

COMPOUND SPATIAL ENTITIES

One contribution of this work is a representation of a compound spatial entity called a footprint. In this work, footprints represent concepts like groups of local units (Figure 8) or a group of red balls on a table. They are constructed by computing the convex hull of a set of constituent regions.

Figure 8: A Freeciv scenario and resulting visualization of footprints. Colors indicate player allegiances. Individual units, e.g. E1 and E2 (left) are represented by spatial regions E1 and E2 (right). Compound entities are constructed by clustering these regions using RCC2 into groups of entities that are connected, e.g. E1 and E2 (right) form a single compound entity.

There are two kinds of footprints. The first, called global footprints, groups a global set spatial entities into one compound entity, e.g. a player's empire is made up of all cities controlled

by the player, no matter how distant. The second class of footprint, called local footprints, effectively clusters entities into groups using a distance metric, resulting in a set of footprints (each cluster becomes a footprint). Figure 8 shows a set of local footprints which are groups of military units, where color indicates the allegiance of each group (i.e. the red player has two groups). Local footprints are constructed by computing networks of entities that are connected. Formally, connection is defined using RCC2 (i.e. the rcc2-connected relation). Recall that RCC2 is a spatial calculus that distinguishes between regions that are connected and disconnected. To compute local footprints, first, a graph is constructed where nodes are spatial regions (e.g. Figure 8, E1 and E2 right) and there is an edge between two nodes when they are connected, e.g. (rcc2-connected E1 E2). Then, each connected component of the graph is a distinct footprint, where nodes of each component correspond with participants of the footprint.

Persistence for Compound Entities

For QSTEM descriptions, one of the main goals is to describe how objects change, e.g. the increasing volume of a group of objects or the decreasing size of a player's civilization. Describing change for these objects requires tracking them across time, which requires specifying how they persist. The main problem is that they are fluid, i.e. they can gain and lose parts. Consider an invading army that loses units over the course of a battle. The question is then, what constitutes the army at each temporal instant? Intuitively, the group at one instant is 'more or less' the same at the next instant. The statement 'more or less' suggests that logical identity is too strong of a constraint, i.e. there are attributes that differ between the two. Individual participants may die, may flee and additional forces may join, etc. This requires some weaker notion of identity. In this work, determining persistence is broken down into two steps. The first step is tracking parts, e.g. a compound entity splits into two entities. The second step applies an

identity criterion to determine which parts persist. Consider two different cases; in one, a single archer flees from a battle, leaving its battalion. In the second, a battalion splits in half where each half defends different locations. Both involve a single battalion splitting into two parts. In the first case, it is natural to say that the battalion persists despite losing a single participant. In the second, determining persistence is not as clear.

Computing persistence happens at each timepoint, after compound entities have been computed. The algorithm, (Algorithm 2, appendix), takes two arguments, the set of compound entities computed at the previous (prev) and current (cur) times (t_{n-1} and t_n in Figure 9). First, a directed graph is constructed where members of prev and cur are source and sink nodes, respectively. Then, arcs in the graph are added between members of cur and prev that share participants, e.g. (CE1→CE4, CE2→CE5, CE3→CE5, CE3→CE6). Next, the connected components are computed for this graph. Intuitively, any number of entities can merge or split at subsequent timepoints; the connected components provide localized descriptions of these phenomena (as well as common cases where all parts are the same). From Figure 9, there would be two connected components, one for the simple case and one for the complex

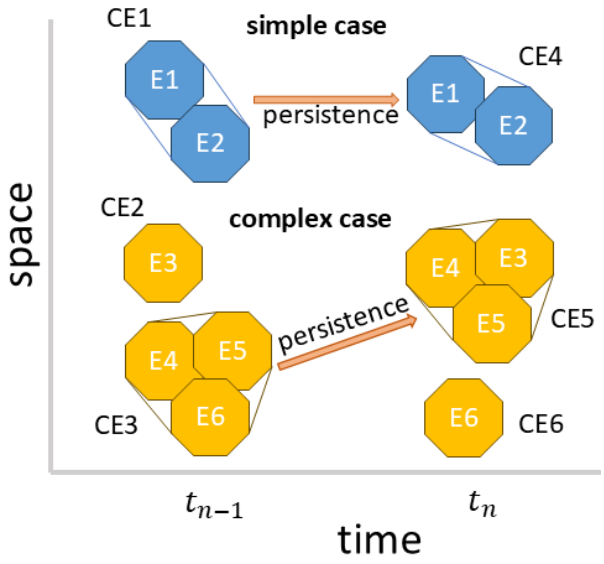


Figure 9: Change for compound entities over time for a simple and complex case. In the simple case, members of CE1 are the same as CE4, so the entity CE1 persists as CE4. In the complex case, merging and splitting occurs across subsequent timepoints. In the complex case, CE3 persists, as CE5, while CE2 is removed, and CE6 appears.

case. The second step is to apply a persistence criterion for each connected component. In this work, the largest entity of each connected component persists, e.g. CE3 persists as CE5. This is accomplished by persisting the name, i.e. CE5 keeps the name of CE3. In the simple case, CE1 persists as CE4 because its constituent entities did not change.

In general, questions of identity across time are complex and involve many factors. Problems relating to persistence have been addressed in philosophy, e.g. The Ship of Theseus. A thorough treatment for a naïve theory likely requires a rich ontology, depending on objects, tasks, etc. The definition used in this work is simple but works in both domains. Situations where this definition might fail include cases where compound entities split into evenly sized entities.

So far, I've described the spatial entities and relations that make up QSTEM cases, and how to track them across time. Next, I introduce a temporal representation that describes how these entities change. Qualitative temporal descriptions serve two main purposes:

- They describe what is happening, e.g. the decreasing height of a falling object, in a domain-independent way.
- They provide a means for determining the temporal bounds of episodes, i.e. segmentation; episodes last as long as some qualitative state holds.

The next section describes the temporal aspect of QSTEM, and then how it is used to build cases.

QUALITATIVE DESCRIPTIONS OF CHANGE

For QSTEM, descriptions of change are reified as bounded temporal intervals over which some qualitative state holds, e.g. a period of an object's decreasing height, or a period where a

group of military units are growing larger. QSTEM uses three qualitative descriptions of quantity: the magnitude sign, derivative sign, and change or quiescence encodings. The magnitude sign (MS) encoding describes the sign of a quantity at some time, i.e. negative, zero, or positive.

Magnitude Sign Encoding: $f(x_t) = sign(x_t)$

For strictly positive quantities, this results in the some/none distinction e.g. there are some invaders in one's territory. The derivative sign encoding (DS) is defined as the sign of the derivative of a quantity over successive instants.

Derivative Sign Encoding:

$f(x_t, x_{t-1}) = sign(x_t - x_{t-1})$

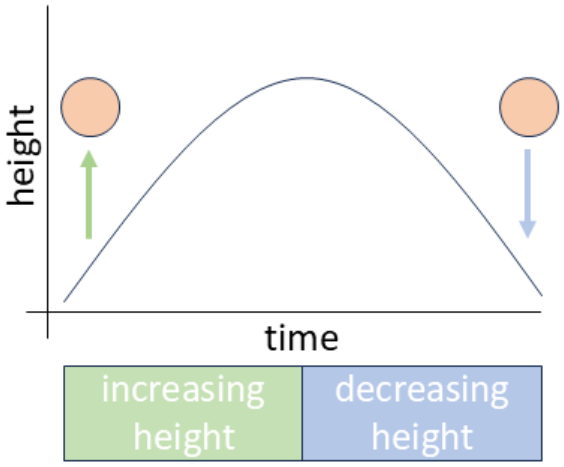


Figure 10: Two temporal intervals describing periods of increasing (green) and decreasing (blue) height of a ball.

For example, Figure 10 shows how the height of a thrown ball using the DS encoding results in two intervals, one where the ball's height is increasing, and one where its height is decreasing. The change or quiescence encoding (CQ) describes whether a quantity is constant or changing over successive timepoints. For example, applying CQ to an object's position results in periods where an object is moving or stationary.

Change or Quiescence Encoding: $f(x_t, x_{t-1}) = sign(abs(x_t - x_{t-1}))$

The resulting temporal intervals are reified as Davidsonian Schemas, which I describe next.

EVENT SCHEMAS

Intervals are represented as Davidsonian event schemas, i.e. the interval is reified and schema predicates describe changing aspects of the interval. These predicates are binary relations corresponding to the qualitative encodings defined above. For the DS encoding, the set is (nonIncreasingIn, constantIn, and nonDecreasingIn). For example,

(nonDecreasingIn I1 (Height Object1))

indicates that the height of Object1 is monotonically nondecreasing over the interval I1. Relations for the magnitude sign encoding are: (negativeIn, zeroIn, and someIn). For example,

(someIn I2 (Gold USA))

indicates that the amount of gold owned by the USA is strictly positive over the interval I2. For the CQ encoding, the corresponding relations are (constantIn and changingIn). An example is

(changingIn I3 (Position Object2)),

which says that Object2 is moving (its position is changing) over interval I3.

CASE CONSTRUCTION

Given a set of intervals, the next step is to reify the temporal structure of an episode. The idea is to describe relations that are important for learning, e.g. 'the number of defensive units was zero before the city was overrun' describes a failure that could be avoided. To do this, Allen's Interval Algebra (AIA) (Allen, 1983) is used. AIA defines thirteen temporal relations between one-piece intervals, shown in Figure 11, which correspond to intuitive notions of time like one interval starting after another ends (meets) or one interval being contained in another (during).

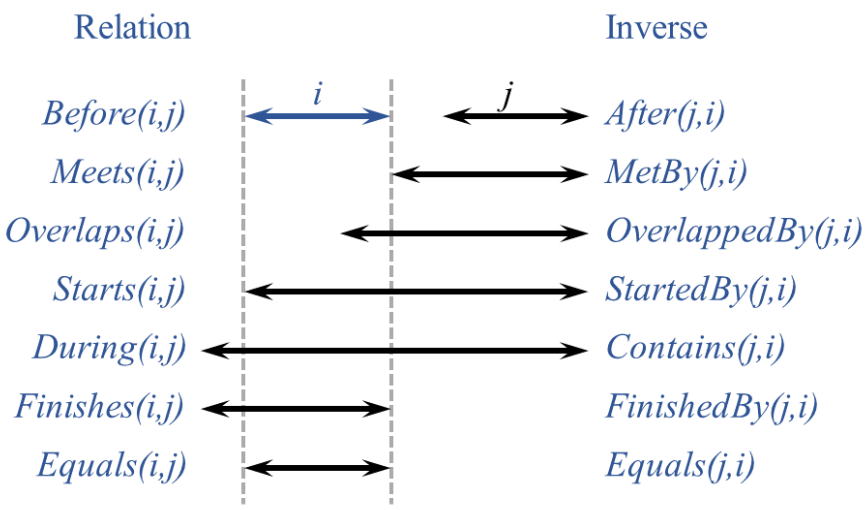


Figure 11: The 13 relations of Allen's Interval Algebra

For building the event structure (Algorithm 3, Appendix A), a set of top-level event intervals are chosen which describe some important aspect of the event. For example, for learning events with AILEEN, AILEEN's hand movement is used, e.g. intervals where the hand is stationary or moving. Then, each entity that participates in the event is described by an object history, which has its own top-level object intervals. For example, in the robotics domain, top-level object intervals are periods where the object is moving or stationary. These are related to other descriptions of the object, e.g. a period of decreasing height. Figure 12 shows an example with two participant objects (entity1 and entity2), but an event can have an arbitrary number of participants, and determining participants is different for each domain. The criteria used in each domain is covered in turn.

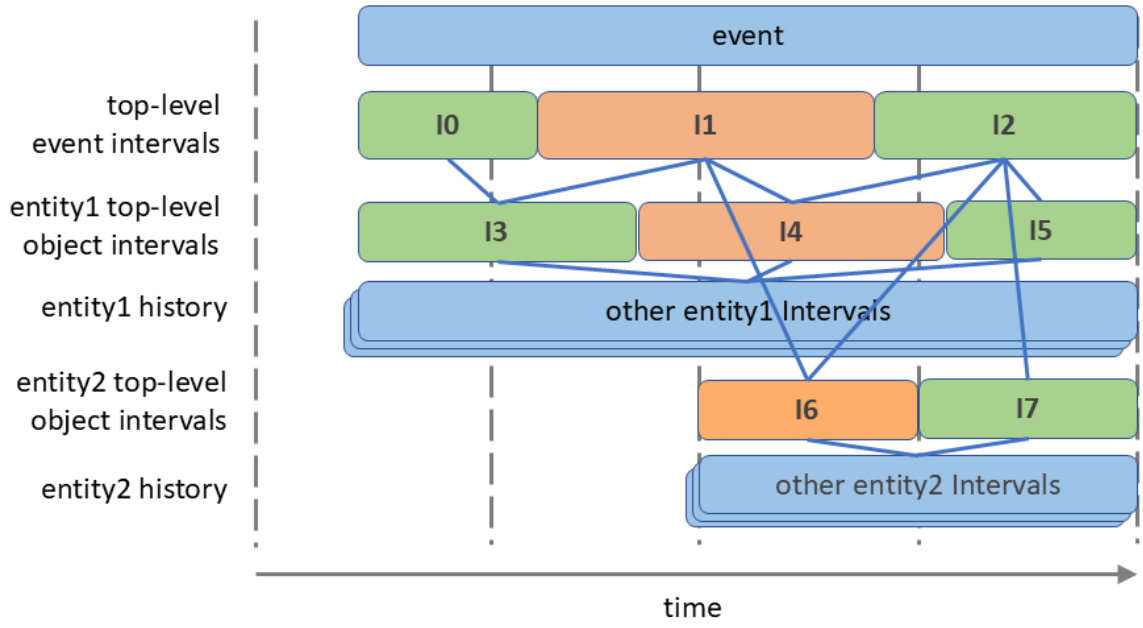


Figure 12: Event structure used in both domains. Using AIA, a set of top-level event intervals are related to connected top-level object intervals for each entity, e.g. (starts I0 I3). Then, top-level object intervals are related to other connected intervals that also describe that object, e.g. interval I3 is related to 'other entity1 intervals' that are connected. Two intervals are connected if the precedes relation (or precededBy) does not hold between them. I0 and I3 are connected (starts I0 I3), whereas I2 and I3 are not (precedes I3, I2). Blue lines indicate connected intervals.

Then, AIA is used to describe relations between intervals. The general event structure for QSTEM (Figure 12) is as follows. Each top-level event interval is related to each *connected* top-level object interval, e.g. (starts I0 I3). Then, each top-level object interval is related to other connected intervals that also describe the object, e.g. (starts I3 <?some-entity-1-interval>). Here, connected is defined using AIA; two intervals are temporally connected unless either precedes or precededBy holds between them, e.g. I0 and I3 are connected, I0 and I4 are not. Then contiguous intervals of the same type are related, e.g. (meets I0 I1), (meets I1, I2). The final step is to reify type-statements and attributes of participants. Type statements are described by Cyc[5]'s isa predicate, e.g. (isa object1 Freeciv-City), meaning object1 is an instance of the concept of a

[5] https://cyc.com (Visited on 4/1/2025)

Freeciv city. Other relations are also reified, e.g. (cityOwner FC-City-Chicago FC-Player-GeorgeWashington) meaning Chicago's allegiance is to George Washington.

ENCODING TEMPORAL INTERVALS

The encoding of qualitative intervals depends on periodically sampling quantities so that their changing properties can be monitored. This requires defining a set of quantities in each domain and deciding when to sample. Examples of quantity types in Freeciv are the area of a player's civilization and the number of enemies in a battle. In AILEEN, examples are the volume of a group of objects and distances between entities. The sampling process is different in each domain, so each is covered in turn. Each quantity type is associated with one or more qualitative encodings. For each (quantity-type, encoding) pair, a series of contiguous intervals is generated from the set of sampled fluents. For example, a falling object's height might be given by a series of fluents specified by (time, value) pairs, e.g. ((0s 1m) (.1s .9m) (.2s .8m)). For the DS encoding, this series would be represented by a single Davidsonian relation (nonIncreasingIn I0 (Height Object1)). Algorithm 1 shows pseudocode for computing the derivative sign encoding for a quantity. This process uses smoothing, which is useful for reducing the amount of noise in

Algorithm 1: Qualitative Encoding for DS

Input: **qvals**: $[(t_n, val_n)\ (t_{n-1}, val_{n-1})\ \ldots]$
a list of quantities, latest first
ints: existing intervals, latest first
Parameter: **mc**: minimum change
τ: minimum time for constant
Output: Possibly updated ints

```
1:   Let dt= qvals[0].t - qvals[1].t
2:   Let dv= (qvals[0].val – qvals[1].val) / dt
3:   if abs(dv) ≤ mc then
4:       dv = 0.
5:   end if
6:   Let enc= sign(dv)
7:   Let lc= end time of last change from qvals
8:   if (enc ≠ ints[0].enc and enc ≠ 0) or
       (enc==0 and qvals[0].t – lc ≥ τ) then
9:       ints[0].end_time = qvals[0].t
11:      ints[0].enc = enc
12:      Let newint= Interval()
13:      newint.start = qvals[0].t
14:      push newint onto ints
15:  end if
16:  return ints
```

event descriptions. Here it reduces the number of participant intervals in a case by removing those where change falls below the minimum change (mc) parameter. Additionally, the minimum quiescence (τ) parameter indicates the amount of time that must pass for a quantity to be considered constant. This algorithm is incremental, and is run for each (quantity-type, encoding-type) pair at each new timepoint. The algorithm for the CQ encoding is the same as Algorithm 1, except that the absolute value of the sign is taken on line 6. The conditional on line 8 determines whether the interval type has changed, e.g. from increasing to decreasing. If so, the previous interval is closed and a new interval is instantiated. There are two conditions; the derivative has changed and the new state is increasing or decreasing, or a constant interval has lasted longer than the τ parameter.

To reiterate, QSTEM is a domain-general framework for describing the changing properties of spatial things using histories. The next chapter describes a set of experiments where an agent uses QSTEM descriptions to improve its gameplay performance in Freeciv.

CHAPTER 3: EXPERIMENTS IN FREECIV

This chapter describes two experiments that evaluate episodic memory for decision making in Freeciv. In the first experiment, the agent learns how to make decisions from recorded traces of a human playing the game. The idea is that, given a similar spatiotemporal context, the agent should make the same decisions as the human would in that situation. In the second experiment, the agent improves its gameplay performance by incrementally learning from its own experience. The agent records traces of military battles and uses those experiences to refine a domain model that is used for decision making. I begin by describing how cases are constructed followed by the two experiments.

Spatial Representations

The basic spatial entities in Freeciv are tiles, which are laid out in a grid. For QSTEM, tiles are represented as regions defined by their four corners. The default tile set configuration is used, where the map wraps in one dimension, resulting in a cylindrical coordinate system. This work uses four kinds of footprints for Freeciv: unit, national, unit-group, and city (Table 1). City and unit footprints are built out of map tiles. National and unit group footprints are built out of city and unit footprints, respectively. Unit footprints are defined as the region defined by its current tile plus the max distance that type of unit can move in a single turn. For most units, e.g. settlers, this is the region defined by the unit's current tile and the eight tiles immediately surrounding that one. Other units like explorers have more move points per turn, resulting in a larger region. Formally, this is the bounded region defined by the set of tiles for which:

$$ChebyshevDistance\left(tile_{unit,}\ tile_{i,j}\right) \leq \text{maxUnitTypeMovePoints} \quad (1)$$

City footprints consist of the convex hull of twenty-one tiles that are within a city region. This is defined in the game as tiles that contribute resources to a city, e.g. food and shield production. National footprints are defined as the convex hull of a player's city footprints. The final footprint type, the unit group footprint, is defined as the convex hull of *spatially local* unit footprints owned by the same player. Spatial locality is defined as a binary relation between two regions, using the region connection calculus 2 (RCC2). Recall that RCC2 has two relations: RCC2-connected and RCC2-disconnected. For QSTEM, two regions are spatially local if the relation RCC2-connected holds between them.

Table 1: Footprint definitions

Footprint Type	Definition
city	Convex hull of 21 tiles that are in the city region
unit	Region defined by Equation 1
national	Convex hull of player's city footprints
unit group	Convex hull of spatially local units of same type

The four region types in Table 1 form the basis for spatial entities in a Freeciv case. The next section describes how footprints are selected for case construction, and then how change is represented for these entities.

Case Construction

The first step for case construction is to determine what entities should be included in the description. In both Freeciv experiments, the focus is on decision making for cities, so entities that are spatially local to the relevant city are included. Case construction starts with the city's footprint (Figure 13, starred). For the first experiment, all spatially local entities at the time of case construction are included (Figure 13, numbered). For representing battles in the second experiment, a type restriction is used as well (only military unit-group footprints). Next, a history for each entity is reified, which contributes to the spatiotemporally larger overall episode history; the idea is that spatial locality provides a means for determining which entities participate in an event, and then change for each of these entities is described as a history. Figure 14 shows an example history for a city footprint. Recall that a history for each entity has a set of top-level object intervals that are related to other intervals that describe the entity. For city footprints, top-level object intervals are the city size quantity with the DS encoding. These kinds of qualitative

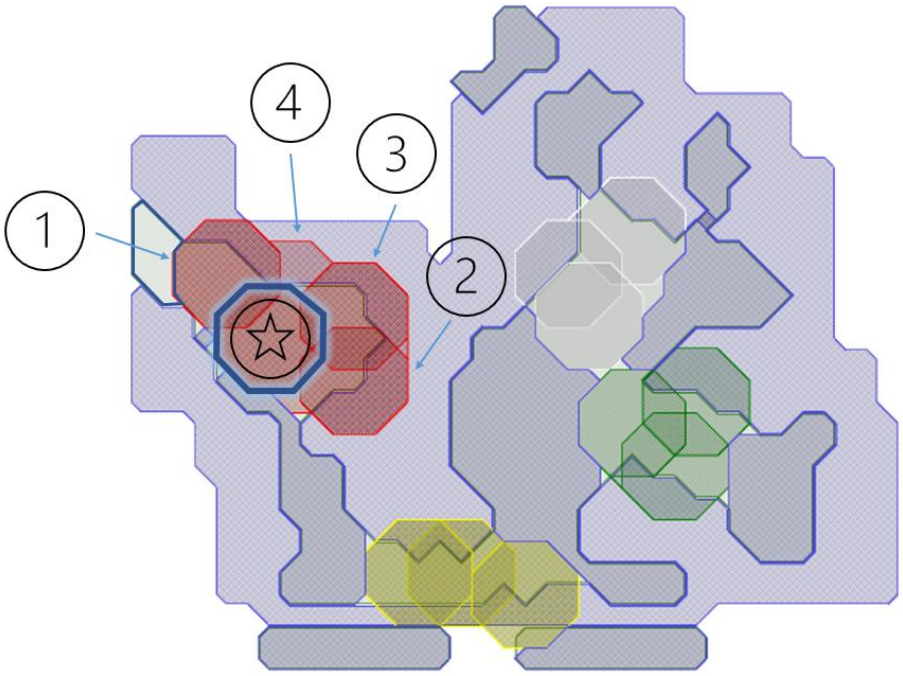


Figure 13: The spatial components of an episode for a city. The starred region depicts the city's footprint. Numbered regions indicate spatially connected entities such as other city footprints (1, 2, and 3) and the player's national footprint (4). Each of these entities contributes a history description to the overall case.

periods are reified as event schemas, e.g. (nonDecreasingIn I1 (Size Chicago)), or the number of defenders stationed in Chicago, e.g. (someIn I3 (Defenders Chicago)). Then, these intervals are related to each other using Allen's Interval Algebra. Top-level object intervals (Figure 14, interval I1) are related to other intervals that are temporally connected, e.g. (overlappedBy I1 I3). Additionally, succession relations are reified for intervals of the same type, e.g. (meets I3 I4), (meets I6 I7). In some cases, the Allen Interval relations between two intervals are ambiguous, because these relations depend on knowing the endpoint of at least one of them. If one interval in question is complete, it may be possible to assign an AIA relation with an interval that is incomplete. If not, or if both are incomplete, the set of possible relations is reduced to (startsBefore, startsAfter, and startsAtSameTime). It follows that relations between current (i.e.

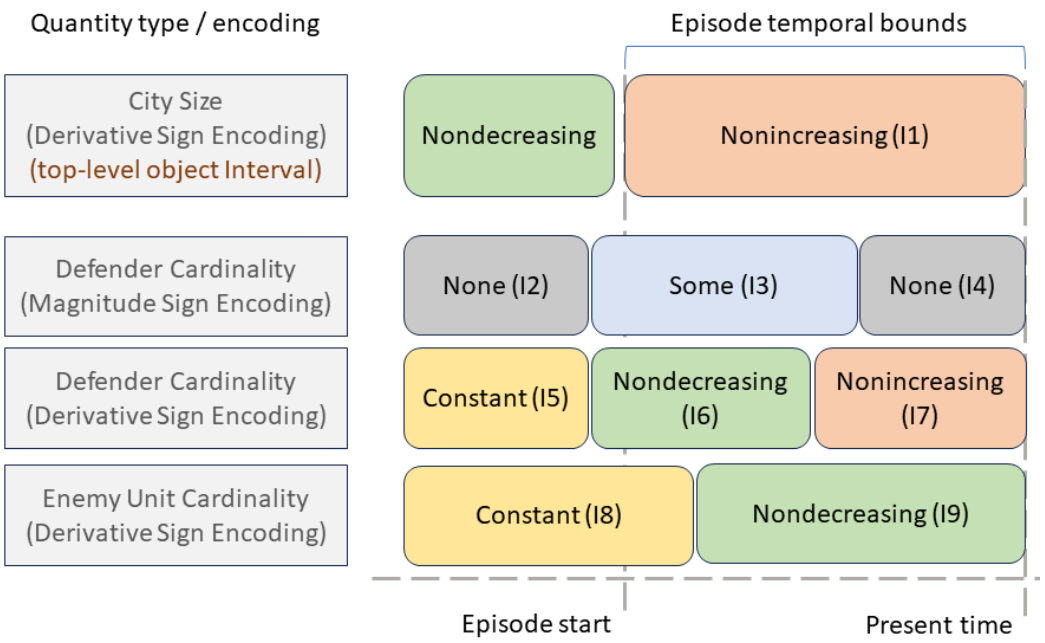


Figure 14: A history for a Freeciv city footprint. The top-level object interval I1 determines the temporal scope of the episode. Other intervals that are temporally connected are reified and related to I1 (I3, I4, I6, I7, I8, I9), i.e. (aia-overlapsInverse I1 I3), (aia-finishedBy I1 I4), etc. Additionally, succession relations are reified for intervals of the same type. For example, (aia-meets I3 I4), (aia-meets I6 I7), (aia-meets I8 I9). This figure depicts the history for a single entity, the city. A full case may additionally include histories for other participant entities (see Figure 13, numbered).

what is happening at the time of encoding) qualitative intervals (e.g. I1, I4) will always devolve to one of these three relations.

The final step for case construction is reifying type statements. The type of each entity is given by isa statements, e.g. (isa ChicagoFootprint CityFootprint), (isa Unit1 Freeciv-Settlers). A full list is given in Appendix A.

Recording Fluents

Ultimately, qualitative descriptions of change must be computed from sampled fluents. This requires deciding when to sample. The agent represents time using an event counter based on event packets from the underlying Freeciv game server. For example, when a unit attacks

another, two packets are sent. The first is a packet describing the attack. The second is a packet describing the removal of the loser. Each packet represents an event at some timepoint. Fluent sampling happens at the beginning of certain event types, i.e. the start of a new turn, the introduction or removal of some entity, and unit movement. Quantity definitions for Freeciv take the form

(quantityName <?name> (<?quantity> <?set-intension>)).

This says that the quantity <?quantity> should be recorded for all members of the set denoted by <?set-intension>, e.g. Area for <?quantity> and all unit-group footprints for <?set-intension>. To sample quantities at some time, first, the set extension is computed, e.g. the set of all unit group footprints. For sampling, the associated quantity for each (here, Area) is queried. Figure 15 shows the definition for the area of national footprints. MeasurableQuantityFn is a logical function that denotes the quantity defined by its single argument. The full list of these definitions for Freeciv is provided in the appendix.

```
(quantityName NationalFootprintArea
 ((MeasurableQuantityFn blobArea)
  (TheSetOf ?fp
   (and (isa ?fp NationalFootprint)))))
```

Figure 15: Quantity definition for the area of national footprints

Computing qualitative temporal intervals from these fluents uses Algorithm 1, described previously. One goal of these representations is to provide intuitive notions of change. It is reasonable to describe the initial growth period of one's empire as lasting for the first one hundred turns or so, but there are many periods of quiescence in between. That is because it is only possible to build a city once every few turns, i.e. there might be five turns where a player's civilization size does not change. Because of this, temporal intervals corresponding to actual change are liable to be quite short. For this reason, algorithm 1 takes a parameter τ that dictates

the minimum time for a qualitative state to be quiescent, i.e. hysteresis is applied. In other words, if a quantity has not changed for longer than τ, then it is considered constant. I use a value of 50 for τ for all Freeciv work. Recall that there is also a minimum change parameter, which is set to 0 for Freeciv experiments.

So far I've described how history-based events are represented in Freeciv. Next, I describe the first Freeciv experiment.

EXPERIMENT ONE: LEARNING GAMEPLAY FROM OBSERVATION

Recalling past experience can assist an agent in deciding what actions to take. For the first experiment, first reported in (Hancock and Forbus, 2021), the agent learns how to make production decisions from watching a human play. Production decisions in Freeciv are assigned to cities and are crucial to winning. A city can produce many different resource types. Combat units and city walls are needed to defend cities. Settlers are needed to create new cities. Many other improvements are useful for helping cities grow. There is an inherent tradeoff between growth and defense early in the game. Too much growth leads to underdefended and smaller cities. Too little growth means that a civilization is unable to keep technological pace with its opponents. It's therefore important to make decisions that are based on an agent's circumstances: starting off on an isolated island versus being sandwiched between two other civilizations calls for different strategies. Expanding one's civilization is important early on, but once territory becomes less available, a player's focus should be on defending and improving existing cities. Furthermore, a player may make different decisions when preparing against a siege vs retreating troops. The first two cases can be described by the semantics of RCC8, the last case by describing the changing sizes of spatially local enemy forces using QSTEM's temporal descriptions of change.

Gathering Data and Learning a Model

The basic idea is that the experimental agent first watches a human play the game and records a set of event descriptions (i.e. QSTEM) and associated decisions. The event descriptions describe in a general way what is going on locally around the entity making a decision when a production decision is made. The idea is that these descriptions encode what is salient to the decision. The experimental AI agent then plays in a set of new, unseen scenarios, and makes decisions based on what it learned from watching the human. To collect cases from human gameplay, ten games were played by a single person (the author of this thesis) until turn 100 or the player was eliminated. Elimination occurred in two out of the ten games. The map was revealed for each of the ten scenarios, as opposed to the default setting where the player must explore to reveal the map. The human player has moderate experience playing the game and wins about 50% of games against the "hard" Freeciv built-in AI. A general strategy balancing expansion, defense, and scientific discovery was used. Specialty strategies like obtaining rapture (a specific strategy for rapidly growing the sizes of a player's cities) were not used. Each time the human player made a production decision, a QSTEM case was automatically generated. This resulted in 463 cases total making up the training set. For learning a model, the case order is randomized, and cases are generalized using SAGE. Recall that SAGE builds up generalization pools from sets of cases. For production decisions, the task is to learn which resource a city should produce. Options include coinage, military units, workers, and city infrastructure improvements. A gpool is created for each resource type that the human player chose to produce. For example, given a new training example where the human decided to produce settlers, the case describing the spatiotemporal context for the city making the decision is generalized into the *settler* gpool. If the gpool is empty, then the case is added as an outlier. Otherwise, the case is

compared against existing outliers and generalizations in the gpool. SAGE returns the top three matches to these entities, defined by a similarity score. If the top similarity score is above the SAGE assimilation threshold, then the case is merged with the corresponding entity. Otherwise, the case is added as an outlier.

The SAGE model, consisting of gpools for each possible decision type made by the human, is then used by the AI player for decision making. When it comes time for a city to make a production decision, first, a QSTEM case is constructed for the city. Then, that case is used as a probe and the gpool which contained the most similar generalization or outlier is returned. The corresponding resource type to that gpool is then chosen as the next production decision for the city. This process is invoked when a city is first founded, and also when it has completed a previous production item.

For this experiment, there are three main hypotheses. The first is that QSTEM will improve the agent's gameplay performance. There is a chance that QSTEM will not encode salient aspects of gameplay, or that the cases include too much knowledge and therefore it is hard to separate signal from noise. The second is that QSTEM will enable the AI to make more human-like decisions. The third hypothesis is that footprints facilitate learning because they capture cases at a salient spatiotemporal grain size.

To test these hypotheses, an agent that learns using QSTEM episodes (QSTEM condition) is tested against a baseline where episodic encoding does not include extended spatial or temporal representations (baseline condition). For the baseline condition, the agent uses the same analogical learning processes as the spatiotemporal agent and learns from the same human replays. However, it does not include spatiotemporal facts in its representations. Instead, it uses

the same entity attributes as the spatiotemporal case; this consists of attribute facts for the city making the production decision, as well as the player (Table 6, appendix).

There are many other choices to be made in Freeciv besides production decisions. Decisions about where to build new cities, improve terrain, build roads, what technologies to research, etc. are all required to effectively grow one's civilization. These decisions are handled in the same way for the baseline and spatiotemporal agents. A goal-based agent is used that leverages a learned qualitative model of Freeciv for decision making (Forbus and Hinrichs, 2019). The agent makes investment decisions about continuous quantities (e.g. should it increase the rate of science production in this city) and only considers type-level goal activation as its decision criterion (see Forbus and Hinrichs (2019) for details).

The first hypothesis is that history-based representations will help with decision making and result in better gameplay. Each condition was tested in twenty new scenarios and statistics for the games were recorded. The maps in these scenarios were partially revealed in the same manner as the scenarios played by the human; the entire continent on which the agent started was unhidden. Success in Freeciv is complicated; there is no single metric to gauge performance. Civilization size, average city size, gold, science achievement, and many other factors go into determining an agent's performance. Overall, surviving later in the game corresponds to better performance. However, it may be possible to survive by focusing only on defense while ignoring other aspects of one's civilization. For this reason, two metrics are used: the number of cities that have survived and the amount of gold accumulated for the player. Evidence from watching the QSTEM agent play shows that it is making contextually relevant decisions. Border cities

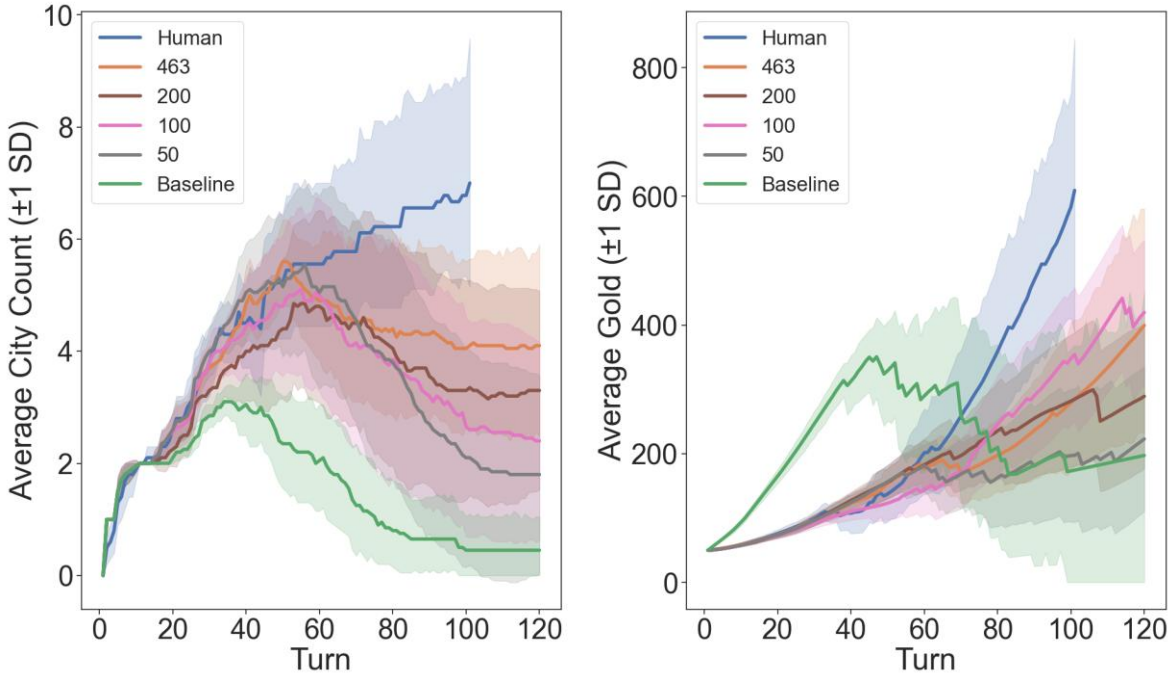


Figure 16: Experimental comparing varying numbers of training cases for QSTEM representations against the human and baseline conditions. The graph on the left shows city size over time and the graph on the right shows accumulated gold for the player, both averaged across twenty scenarios.

adjacent to enemies tend to specialize in defense, whereas interior cities are more likely to produce settlers and gold. In Figure 16, all conditions are increasing their territory size at turn 50. Enemies arrive at around turn 70, and there is a correlation between the number of training examples and the agent's ability to defend itself. The baseline condition begins to specialize its economic production early on (Figure 16, average gold) but suffers later because it can't defend itself. Both results are significant ($p < .05$ using student's paired t-test). Additionally, Figure 16 shows the outcomes of varying the number of training cases. The results indicate that more examples enable better decision making, which allows civilizations to survive into later turns. With only 50 examples, there is improvement over the baseline, while 463 examples show the best performance. The 100-case condition has more gold at turn 120 than the 463-case condition.

This may be attributed to the model for producing coinage, which contains four examples. With small amounts of training data, the resulting model is unlikely to reflect the true distribution. Indeed upon inspection, the model contains facts describing periods of increased economic growth (which may cause a feedback loop) and increasing enemy attackers, which arrive around turn 70 when the 100-case condition begins increasing its economic output. In comparison with the human's performance, the QSTEM condition averages 4.4 cities vs 6.1 for the human at turn 100. Average gold at turn 100 is 279 for the spatiotemporal condition vs 532 for the human, so there is still room for improvement.

Figure 17 shows histograms for the kinds of production decisions made over time for three different experimental conditions. For each of the three conditions shown, decisions are grouped into twenty turn intervals in the game. Y values indicate the relative frequency of each decision type within an interval.

The baseline focuses on growth and economic development early in the game, and lack of defensive infrastructure leaves its cities vulnerable to attackers later on. Production of defensive units for the baseline player is reactive, due to the nature of goal instantiation depending on ground instances of attackers. This lack of foresight results in its poor performance. The QSTEM condition more closely follows the decisions of the human. Settlers are built in the beginning in order to grow the civilization, mixed with defensive warriors and later stronger archers. The production of archers in the QSTEM condition from turns 40-60 show that the AI begins to focus on defense, helping to protect cities as they come under attack around turn 60.

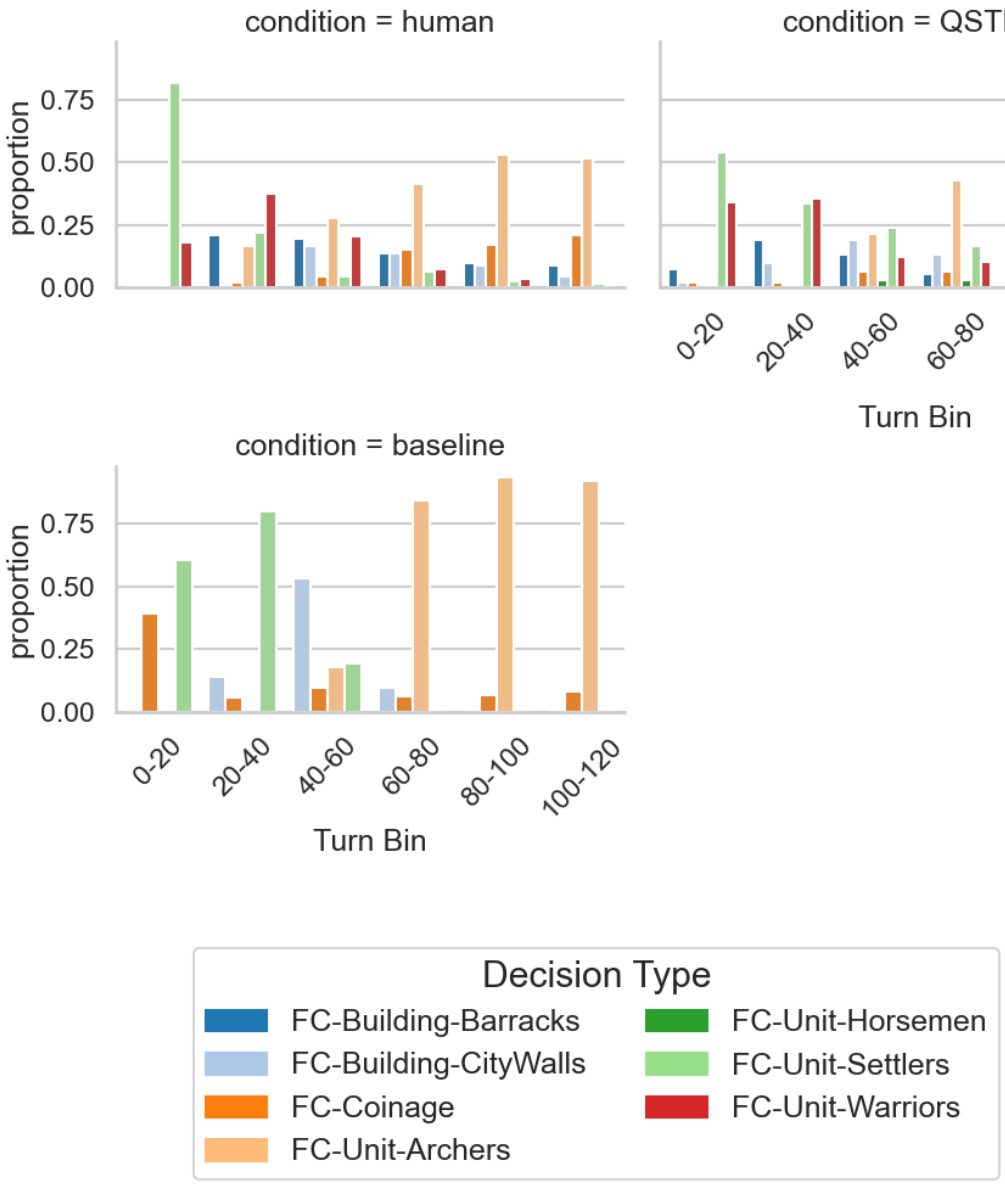


Figure 17: Production decision types over time for the human, QSTEM, and baseline conditions. The QSTEM condition uses the full 463 QSTEM cases. Decisions are binned into twenty turn intervals, with y values indicating the relative proportion of each decision within each bin.

The second hypothesis is that QSTEM will help the AI player make more human-like decisions. To assess this, the QSTEM agent's decisions are compared against the human player's and the baseline model. Each decision made during gameplay is binned into fixed-width temporal intervals to track decision distributions over time (Figure 17). For each time bin, a

probability distribution is computed for relative decision types. These distributions are generated for each condition: the human player, 463-case agent, and the baseline agent. Jensen-Shannon Divergence (JSD) is used to quantify similarity, which provides a symmetric, bounded measure of dissimilarity between probability distributions. For example, the Jensen Shannon Divergence can be computed between the human and baseline decision distributions for the 0-20 turn bin. This metric will be low if similar decisions were made early in the game, e.g. more settlers were produced. To evaluate the hypothesis that the QSTEM condition is significantly more similar to the human player's decisions than the baseline, JSD is computed for each bin between the human and QSTEM conditions and the human and baseline conditions. The resulting quantities are tested for significance using a paired t-test. This analysis showed that the full QSTEM condition is significantly more similar to the human player than the baseline condition (p=.007), implying that QSTEM helps the AI agent make more human-like decisions.

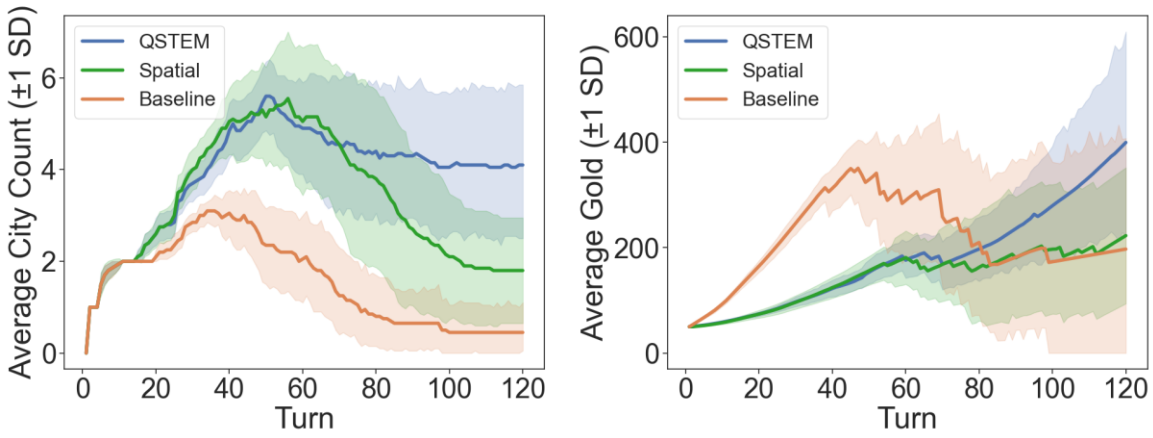


Figure 18: Average city size and gold across twenty games for baseline, spatial, and spatiotemporal conditions. The spatial condition significantly improved over the baseline for number of cities, and the spatiotemporal condition significantly improved over the spatial condition for both.

Figure 18 shows how both the spatial and temporal descriptions in a case contribute to gameplay performance. The baseline condition is the same as the previously reported baseline.

The spatial condition uses the same spatial descriptions and relations as the previous QSTEM condition, but no temporal descriptions. The spatiotemporal condition uses the full QSTEM representations. This experiment uses the same scenarios and settings as the previous experiment. The results indicate that spatial and temporal representations each contribute useful knowledge. For civilization size, the spatial condition significantly improves over the baseline, and the spatiotemporal condition significantly improves over the spatial condition ($p < .05$ using a paired t-test). For gold, spatial representations alone do not improve the agent's performance. Both the QSTEM and spatial conditions maintain roughly the same number of cities until turn 70, after which the spatial condition begins to lose more cities to attackers. A large civilization is key to maintaining economic output, and so a smaller civilization will produce less gold, all else being equal. The baseline and spatial conditions have roughly the same amount of gold at turn 120 due to the overemphasis on economic output by the baseline early on.

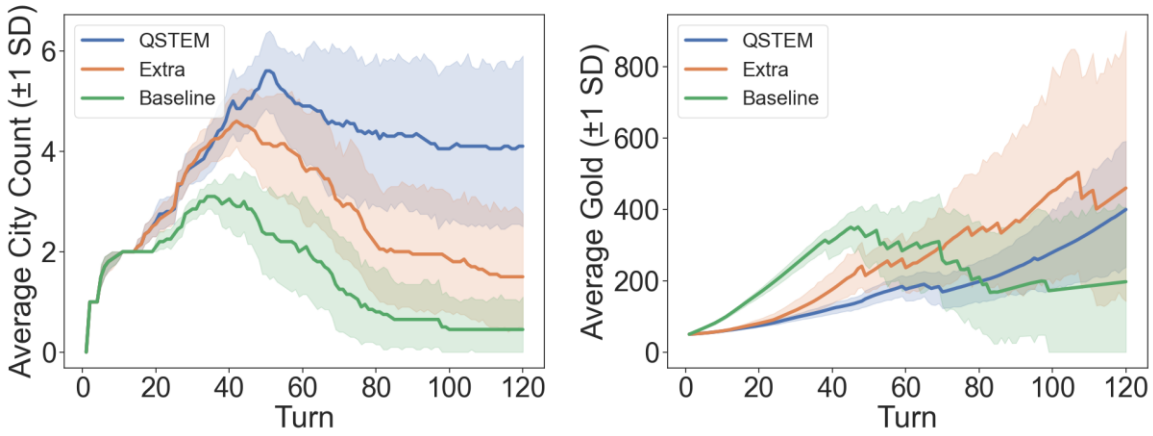


Figure 19: Average number of cities and gold for three spatial conditions. In the extra condition, cases include entities that are local to the city making a decision, as well as entities that are local to those entities. The baseline condition is the same as previously, and the standard condition is the previous QSTEM condition.

Finally the third hypothesis for experiment one is that QSTEM captures events at a useful grain size. Figure 19 shows the effect of varying the spatial extent of episodes on gameplay

performance. The previous baseline and spatiotemporal conditions are compared against the *extra* condition, where episodes describe a larger area of the map. Cases in the extra condition include spatial entities local to the city footprint, as well as spatial entities local to those entities. The results show that including extra spatial entities in episodic cases negatively affects civilization size but is better than no spatiotemporal knowledge for the baseline. Interestingly, the extra space condition produces more gold than the episodic full condition. This is due to an overemphasis on economic development early on, while growth suffers. This can be seen early on, where the extra condition's economy grows faster than the QSTEM condition, around turn 25. This is also around the time when the number of cities in the extra condition begins to fall behind the QSTEM condition. For the QSTEM condition, the generalization for producing coinage contains the temporal fact (startsAfter (increasing civsize) (some citydefenders)). In other words, coinage should be produced when a city is properly defended. The extra condition on the other hand, contains the fact (overlapped (increasing civsize) (zero citydefenders)), or "coinage should be produced in a period of civ expansion overlapped by a period of no defenders in a city). This condition is true the second turn after any city is built. This may be explained by the sparsity of the extra space condition cases; they include facts for more cities, all else being equal, and thus are less likely to distinguish between important factors for the spatial and temporal situation of the city making the decision.

FUTURE WORK

The results shown in Figure 19 indicate that there may be an optimization to be made, i.e. there may be a local maxima for performance somewhere around QSTEM representations (i.e. slightly larger or smaller spatial regions might perform better). One formalism that could help is the egg yolk calculus (Cohn and Gotts, 1995) which would allow reasoning about definitely-near

and possibly-near. For example, salience could be learned for definitely-near entities based on gameplay performance, so that they contribute more to retrieval than possibly-near entities.

The full 463-case QSTEM condition still underperforms compared to the human player; I see two main future directions to consider addressing this gap. The first involves extending QSTEM with more kinds of knowledge, e.g. orientation, distance, or configuration calculi. These descriptions could help with decision making for combat, where the current agent is currently lacking. For example, a configuration calculus could help with military unit positioning and maneuvering. Reifying social knowledge could also help with diplomacy. The second future direction involves using QSTEM for other reasoning besides analogical retrieval. One avenue is to show that QSTEM supports other reasoning tasks like planning and abduction. Planning with representations of memory that are automatically segmented might reduce the search space by including only events that are relevant to some task. Representing events at multiple grain sizes may also help with task abstraction.

To summarize, this experiment evaluates the use of QSTEM for learning Freeciv gameplay from a human. Experimental results show that these representations encode knowledge at an appropriate grain size, and that both temporal and spatial components of the descriptions are useful for learning. With only 463 cases, the AI player approaches human performance for two evaluation metrics: the size of one's civilization and the amount of gold accumulated. Additionally, QSTEM helps the AI make more human-like decisions, showing similar production decisions at different times in the game relative to the baseline condition. In the next section, I describe an experiment where the agent uses QSTEM to learn from its own experience.

EXPERIMENT TWO: LEARNING FROM BATTLES IN FREECIV

Experiment two further investigates using histories for learning. The main idea is that histories are useful for episodic learning because important limit points (Forbus, 2019) happen at the temporal boundaries of events. Limit points are significant points in a quantity space where conditions on either side of the point are qualitatively distinct. Histories are useful for scoping temporal bounds (i.e. event segmentation), and I show in turn that segmentation is important for building domain models and therefore for decision making. The focus for this work is military battles; the agent records traces of battles and makes decisions based on learning from failures. The task is to use QSTEM to learn how to defend cities. I show empirically that history-based vs nonextended representations affect how limit points are learned, and that in turn this affects gameplay performance. The results also highlight the relationship between spatial representations and event segmentation, showing that the perception of space affects how time is carved up. This segmentation affects the quantity and quality of resulting descriptions. I also discuss how histories can help support event segmentation because they facilitate commonsense reasoning about event persistence, and segmentation is inherently dependent on persistence.

Figure 20: A battle consisting of three enemy units attacking a player's city Praha.

LEARNING FROM PAST EXPERIENCE

I begin by motivating the relationship between event perception, i.e. determining the spatiotemporal bounds of an event, and learning from the resulting descriptions. Consider the

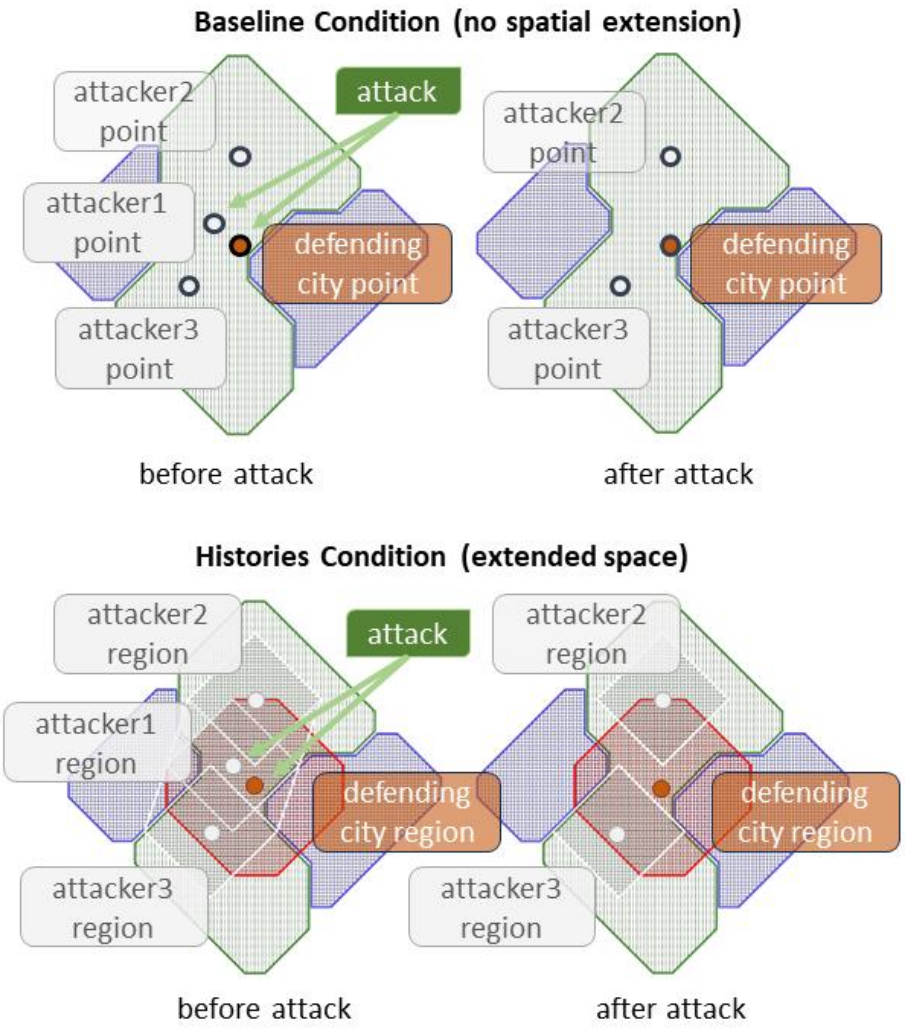


Figure 21: Before and after representation of an attack. Assume that attacker1 loses an attack on the defending city and is removed from the game. For the baseline condition, the battle from the perspective of the city is over because there are no longer any spatially local enemy units, i.e. at the previous location of attacker1 or the defending city, as defined by the given persistence criterion for battles. The same scenario for the histories condition is depicted on the bottom, this time with extended spatial representations. Attacker1 loses its attack against the city and is removed from the game. However, there remain enemy units that are local to the city because their footprints are connected. In this case, the battle persists until attacker2 and 3 successfully take the city, are destroyed, or flee.

scenario in Figure 20, where a group of units are attacking a city. The hypothesis is that without histories, learning will be harder because localized events like battles cannot be adequately characterized. Assume that attacker1 attacks the defending city Praha, that there is at least one defender inside Praha, and attacker1 loses the attack and is removed from the game. In Figure

21, this scenario is illustrated with and without extended spatial representations. Persistence for many events (e.g. battles) is conditioned in part on the existence of participants. That is, one way a battle ends is when one side has been eliminated. This definition of persistence depends on some spatial context, i.e. the battlefield. For this work, I compare a baseline experimental condition that does not use extended spatial representations against the histories condition that does. In Freeciv, the attack action primitive takes the form:

(doAttack ?attacker ?defenders)

where ?attacker is a single entity and ?defenders is a list of defending entities that are all located on the same tile. For the baseline condition, the battlefield is represented by two points: the tiles of attacker1 and the city Praha. After the attack, since the city is the only remaining entity at either location, the battle is over for the baseline condition. On the other hand, Figure 21 (bottom) depicts the same situation given extended spatial representations. Here, the battle persists after attacker1 has been removed, dependent on some persistence criterion. A natural way to define this using qualitative representations is with RCC2. Recall that RCC2 distinguishes between regions that are connected and disconnected. The relation rcc2-connected provides a notion of locality. In the above example, the battle persists because there are still local opposing units after the initial attack, i.e. the relation (rcc2-connected attacker3 city) holds. The differences in temporal perception for each case then impacts learning. Consider the question of how many military units are needed to defend a city. For this example, the goal is to learn how many defenders are necessary to win a battle. Assume that the scenario depicted in Figure 20 eventually leads to the city being conquered. In the histories condition, the battlefield contains the city footprint and the unit group footprint. Here, the battle persists until the city is taken (assuming the locality conditions hold), resulting in a single battle episode containing all three

attackers as participants. On the other hand, for the baseline condition, the same scenario results in three episodes, corresponding to each attack event. Each describes a battle with one attacker, i.e. the spatiotemporal bounds are more fine-grained. The outcomes of each episode are also different. The first two result in successes for the city, while the third results in a failure. This has implications for learning and reasoning, namely that the baseline condition describes cases that are too localized. An agent could perhaps avoid using these kinds of representations by simply reasoning about all entities that exist in a scenario, i.e. all visible units throughout the world map. This forgoes localized space by simply considering all space. In this case, battles cannot be distinguished from wars. The next section describes background knowledge used by the agent and then how battles are represented.

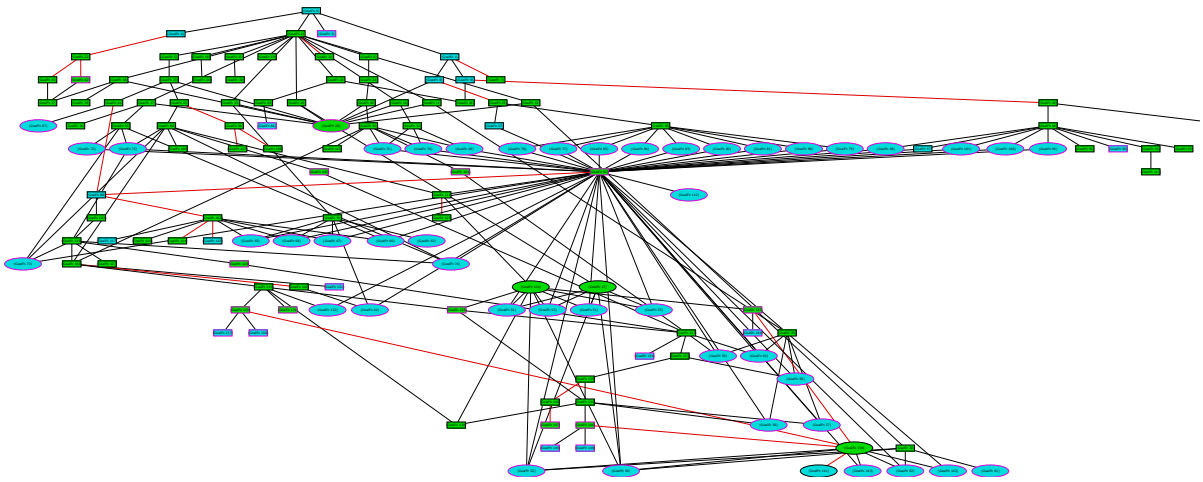


Figure 22: The learned goal network of Freeciv. This graph describes prescriptive constraints over influences, e.g. more gold improves a player's economic system. These constraints can be traced up to the root node (top) which represents winning the game, so static analysis can be run to determine which quantity influences are beneficial and which are detrimental.

Qualitative Event Definitions

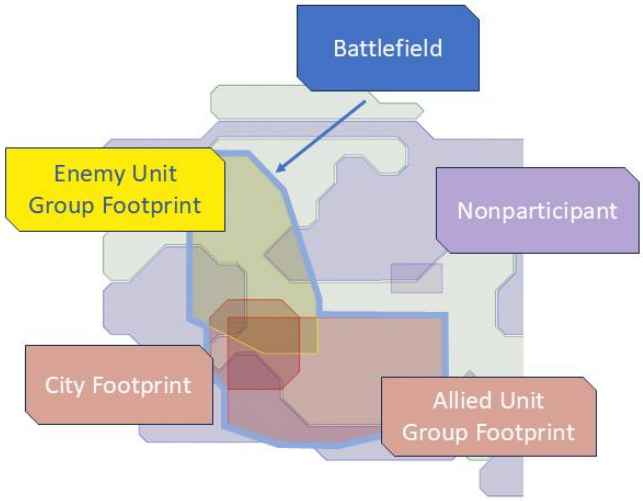
This experiment leverages two types of previously learned domain knowledge about Freeciv. The first is a qualitative causal model and the second is a goal network constructed from this model (Forbus and Hinrichs, 2019; Hinrichs and Forbus, 2016; Hinrichs and Forbus, 2012). The qualitative model describes quantity influences within the game, e.g. increasing the number of defenders in a city increases the overall defensive ability of that city. The goal network has a root node which corresponds to winning the game, which is influenced by all other nodes in the graph (Figure 22). There are two types of goals: achieve goals and quantity constraint goals. Achieve goals describe desired states, e.g. changing the government to a democracy or having city walls in Chicago. Undesired states are denoted by PreventFn, which is a logical function that denotes the prevention goal for the single argument. Quantity constraint goals are prescriptive constraints on quantity change, e.g. minimizing the number of enemy units in a battle. These goals provide no explicit criterion for success, but instead suggest that the goal is decomposable. Figure 23 shows a learned quantity constraint goal to minimize the number of opposing units.

```
(MinimizeFn
  ((MeasurableQuantityFn cardinalityOf)
   (TheSetOf ?unit
    (and (opponentsInConflict (IndexicalFn currentRole) ?opponent)
         (isa ?opponent FreeCiv-Player)
         (sovereignAllegianceOfOrg ?unit ?opponent)
         (isa ?unit FreeCiv-Unit)))))
```

Figure 23: Learned goal to minimize the number of enemy units.

The logical function IndexicalFn resolves to the entity determined by its argument in some context. Here, (IndexicalFn currentRole) denotes the agent's player in the current Freeciv scenario.

For this work, military battles are represented from the viewpoint of a single city, i.e. the event starts when the city is attacked and ends when the city falls or there are no more local enemies. It is possible, however, that fighting has moved on elsewhere. In this sense, there are two histories to consider: the piece of spacetime where the city is participating, and the piece of spacetime that also includes the battle at the new location. In this sense, cases may be constructed deictically or globally. Deictic cases are constructed from the viewpoint of some entity, e.g. a unit fleeing causes the event to end for that unit, even though the battle may continue. A global viewpoint considers the whole piece of spacetime. Here we are concerned with decision making for cities, and thus generate cases from their viewpoints, i.e. a battle lasts as long as there are enemy units that are spatially local to a city or until the city has fallen.




```
(<== (spatiallyContains ?hist-slice ?region)
     (unifies ?hist-slice (AtFn ?event ?time))
     (isa ?event FreeCiv-MilitaryBattle)
     (experiencer ?event ?entity)
     (footprintOf ?entity ?footprint)
     (rcc2-connected ?footprint ?region)
     (or (equals ?footprint ?region)
         (isa ?region UnitGroupFootprint)))
```

Figure 24: Visualization of a battlefield, and rule defining its spatial participants. Participants are the footprint of the deictic city, and any unit group footprint that is local to the deictic city footprint.

Figure 24 shows a visualization of an example battle, and a clause for defining battle participants at some instant. Participants include the footprint of the city, and all unit group footprints that are spatially local to that city footprint. Here, ?hist-slice takes the form (AtFn ?event ?time), where AtFn denotes a slice of the history ?event at some time, i.e. a spatial region. The Neo-Davidsonian role *experiencer* indicates the deictic role of the city. The idea is that given a known battlefield, i.e. ?hist-slice, this rule computes a set of participant entities at some time which are bound to ?region.

```
(goalName WinBattle
  (AchieveFn
   (and (activeEvent ?event)
        (isa ?event FreeCiv-MilitaryBattle)
        (cardinalityOf
         (TheSetOf ?unit
           (and (opponentsInConflict (IndexicalFn currentRole) ?opponent)
                (isa ?opponent FreeCiv-Player)
                (sovereignAllegianceOfOrg ?unit ?opponent)
                (footprintParticipant ?region ?unit)
                (spatiallyContains (AtFn ?event ?cur-time) ?region)
                (isa ?region UnitGroupFootprint)
                (isa ?unit FreeCiv-MilitaryUnit))))
         0))))
```

Figure 25: Achieve goal definition for winning a battle.

Goal representations from the Freeciv qualitative model are used to define start and end conditions for battles. Figure 25 shows a manually generated achieve goal which says that a battle is won when there are no more opposing units on the battlefield. Here, ?event is a history. The predicate activeEvent is true if the event has not ended. This goal leverages the minimization goal from Figure 23 and grounds it into a spatially localized achieve goal. The conditions for this goal provide a mechanism for determining when the event ends, i.e. it is over when the goal has been met. Similarly, the goal to avoid losing battles, which has also been manually generated, is shown in Figure 26. The failure condition is defined by one of two conditions occurring: the destruction of the city, or an opponent conquering it.

```
(goalName AvoidLosingBattle
  (PreventFn
   (and (isa ?event FreeCiv-MilitaryBattle)
        (activeEvent ?event)
        (experiencer ?event ?entity)
        (isa ?entity Freeciv-City)
        (opponentsInConflict (IndexicalFn currentRole) ?opponent)
        (or (sovereignAllegianceOfOrg ?entity ?opponent)
            (destroyed ?entity)))))
```

Figure 26: Goal definition preventing the loss of a battle. In Freeciv, a city may be conquered or destroyed only when all defenders have been eliminated. Conquering results in a change of allegiance to the conqueror. Destruction removes the city from the game. Both cases result in a failure for the city.

The Freeciv agent uses these goal statements to monitor relevant events. At each timestep, the goal condition is queried against the logical environment, i.e. the current Freeciv scenario context. For example, (activeEvent ?event) has as bindings all active events in the current context to ?event.

Given an understanding of goal representations for battles, the next focus is on how an agent can learn limit points. Consider the question of how many military units are needed to defend a city. The causal model encodes the fact that more units are better than fewer, but this must be grounded in some context. The goal for the agent is to learn a model that differentiates between adequate and inadequate defensive states. Producing more units than needed wastes resources while producing fewer than needed leads to undefended cities, and so this kind of learning allows for reasoning, for example, about the minimum number of units capable of defending a city. Assume that the scenario depicted in Figure 20 eventually leads to the city being conquered. For the histories condition, there are three enemy participants in a single battle. For the baseline, there are three battles with one enemy each, where only the final episode is a failure for the city. The implication is that the baseline agent sees only one attacker for the

failure, whereas the histories condition sees three. The baseline event representation is too localized.

Case Construction

Case construction for experiment two uses the same representations and algorithm as in experiment one, so this section provides a brief recap. Recall that battles start when an enemy attacks a city, and end when either the city has been conquered or there are no more local enemy units. During an ongoing battle, participants are the defending city footprint and any spatially local (i.e. rcc2-connected) unit group footprint. Previously, I described how persistence is determined for compound entities. This provides a way to track entities over time as they gain and lose parts. In turn, this allows for qualitative descriptions of change, e.g. a group of units is increasing in size over some interval. This is useful for battles because it allows describing the changing sizes of friendly and enemy units during the battle. Fluents are recorded in the same manner as before, with the same encoding algorithm. Figure 27 shows an example history for a battle. Here, the interval I1 is the top-level event interval. Histories for each participant unit group footprint are also included in the case. Finally, quantity values at the start and end of the event are reified. These statements take the form

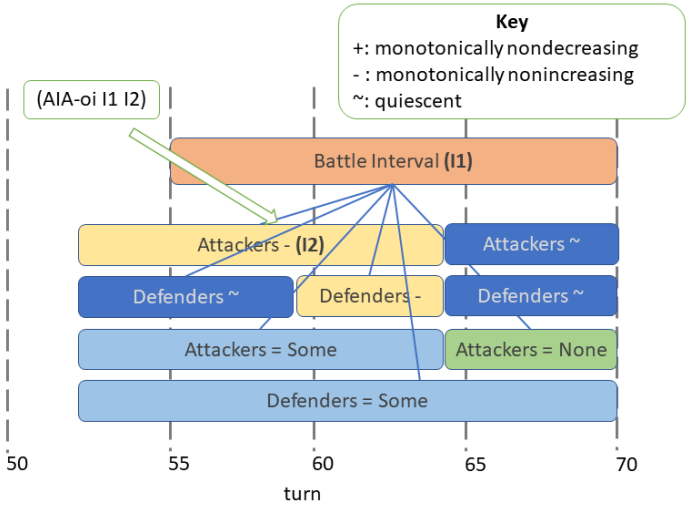


Figure 27: Qualitative temporal intervals for a military battle.

((<?quantity-type> (AtFn <?event> (StartFn <?event>))) <?quantity-value>)

((<?quantity-type> (AtFn <?event> (EndFn <?event>))) <?quantity-value>)

Where StartFn and EndFn denote the start and end time of the battle. For example,

((Attackers (AtFn E1 (StartFn E1)) (UnitCount 2))

((Attackers (AtFn E1 (EndFn E1)) (UnitCount 0))

means "there were two attackers at the beginning of the battle, and none at the end". These statements are reified for every quantity type used in Freeciv cases.

Learning a Model

After each battle has ended, a case is constructed and added to a success or failure gpool, depending on the outcome. Learning is incremental, with cases being generalized in the order that they are experienced[6]. The general process for learning a SAGE model is described next. Assume that an agent experiences a series of failures. For the first battle event, the case is added to the failure gpool as an outlier. For the next example, the case is compared against the existing outlier. If it is sufficiently similar to the first case (as determined by SME's similarity score being above the SAGE assimilation threshold), then the two cases are merged into a single generalization. Otherwise, the second case is also added as an outlier. Learning continues in this matter as each new case is generalized into the model. Each cluster of cases (generalizations) can be seen as a schema for a disjunct for some disjunctive concept. For the battle failure gpool, these clusters describe different conditions where battles were lost, e.g. a prolonged siege vs a quick victory. As part of this learning process, statistics are accumulated for quantities that appear in the structured case descriptions. Recall that these quantities take the form

[6] This online learning means that SAGE models are sensitive to order. We leave order optimization for future work.

((<?quantity-type> (AtFn <?event> <?time>)) <?quantity-value>).

Here, <?quantity-value> is a typed numerical statement, e.g. (UnitCount 3). SAGE automatically accumulates statistical knowledge for typed quantities that appear in structured descriptions, e.g. the mean of the number of attackers on the battlefield at the start of battles. This process proceeds as follows. Assume that a new case describing a battle has been constructed, and that this case is being assimilated into a SAGE generalization. Figure 28 shows an example mapping, i.e. corresponding statements, from a probe case to a gpool generalization. As part of this

Probe Case	Gpool Generalization
((Attackers (AtFn E1 (StartFn E1)) (UnitCount 3))	((Attackers (AtFn (GEFn 0) (StartFn (GEFn 0))) (UnitCount (GEFn 1)))
(isa NYC Freeciv-City)	(isa (GEFn 2) Freeciv-City)
(constantIn I1 (Attackers NYC))	(constantIn (GEFn 3) (Attackers GEFn 4))

Figure 28: Example statements from an incoming probe case (left) and the corresponding SAGE generalization that it has mapped to. Here, GEFn is an abbreviation for GeneralizedEntityFn, which is a logical function that denotes a generalized entity. In this example, (GEFn 1) represents a set of quantities. SAGE automatically accumulates statistics for these quantities, which are used for decision making.

generalization process, recall that mapped terms are replaced by logical functions called generalized entities that denote the lifted type for ground entities, e.g. (GenEntFn <?index>), here abbreviated as GEFn. These reified entities are associated with a history of their values, e.g. for (GEFn 1) (UnitCount 2), (UnitCount 1), (UnitCount 0) from each case in the generalization that has this property. For purely numeric generalized entities, statistics are incrementally accumulated as each new value is seen (here (2, 1, and 0)). The cardinality, minimum, maximum, mean, and sum of squared errors are computed. These statistics are used for decision making, described next.

Decision Making

Decision making is driven by analogical comparison of the agent's current situation to past failures. The idea is to determine if the current spatiotemporal context for a city is similar to a previously lost battle[7]. If this is the case, then a decision is made to improve the city's military system by choosing what resource to produce. This process starts only after an initial growth phase, where the agent is focused on expanding its empire. The growth phase is defined as the initial period where the number of cities is monotonically nondecreasing and there are no enemy unit footprints rcc2-connected to the player's national footprint. After the growth phase has ended is the conquest phase. Making city decisions for this experiment starts in the conquest phase, which allows time for the empire to grow before focusing on defense.

Recall that for learning a model, cases are generated at the end of each battle and added to a SAGE gpool. When a city makes a production decision, a case is generated to describe its current state, hereafter referred to as the probe case. Cases are constructed using the same process as the first experiment, with one additional constraint. Recall that for the first Freeciv experiment, cases include histories for the city making a decision and all spatially local entities. This experiment additionally includes a type constraint when adding local entities, so that only histories for cities and military units are included.

At decision time, the city is not necessarily engaged in battle, and if it is, the end of the battle has not been determined. Because of this, the probe case describes incomplete knowledge with respect to full battles. Recall that for this work, descriptions of battles consist of histories for the reified event (shown in Figure 27) as well as each participant entity. For the probe case,

[7] Recall that analogical comparison is asymmetric, so SIM(A,B) ≠ SIM(B,A). This is controlled for in experiments by only considering a single ordering, i.e. SIM(A,B), where A is the current event model, and B is a SAGE model

histories for participant entities are reified, i.e. the city footprint and any local unit group footprints, but not the event history shown in Figure 27. Thus, there is no top-level event interval, but relevant facts like knowledge about local defenders and local enemy units are still included. The next step for building the probe reifies relevant quantity values. Quantities are sampled at the time of case construction and represented as

((<?quantity-type (AtFn <?event-name> (StartFn <?event-name>)) <?quantity-value>).

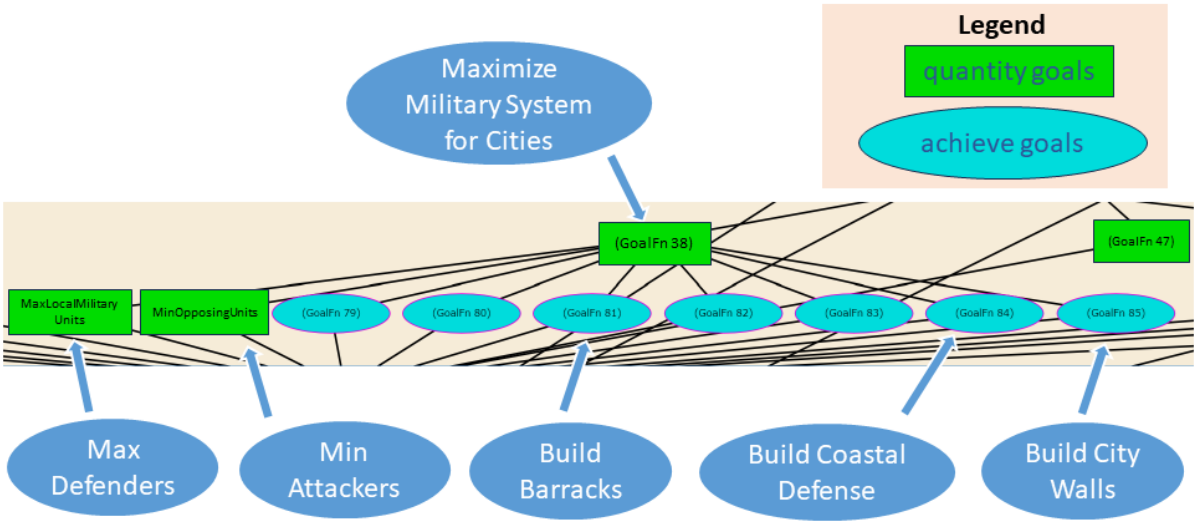


Figure 29: A subgraph of the goal network depicting maximization of city military systems (parent) and its influences.

The idea is that quantity values at the time of construction of the probe case should map to quantity values at the start of battles. In other words, the agent makes an analogical comparison if a battle were to break out at that timepoint.

For decision making (Figure 30; Algorithm 4, Appendix A), an analogical retrieval is performed, comparing this probe against both failure and success gpools. If no mapping is returned, or the retrieval maps to the success gpool (i.e. the agent predicts that the city is

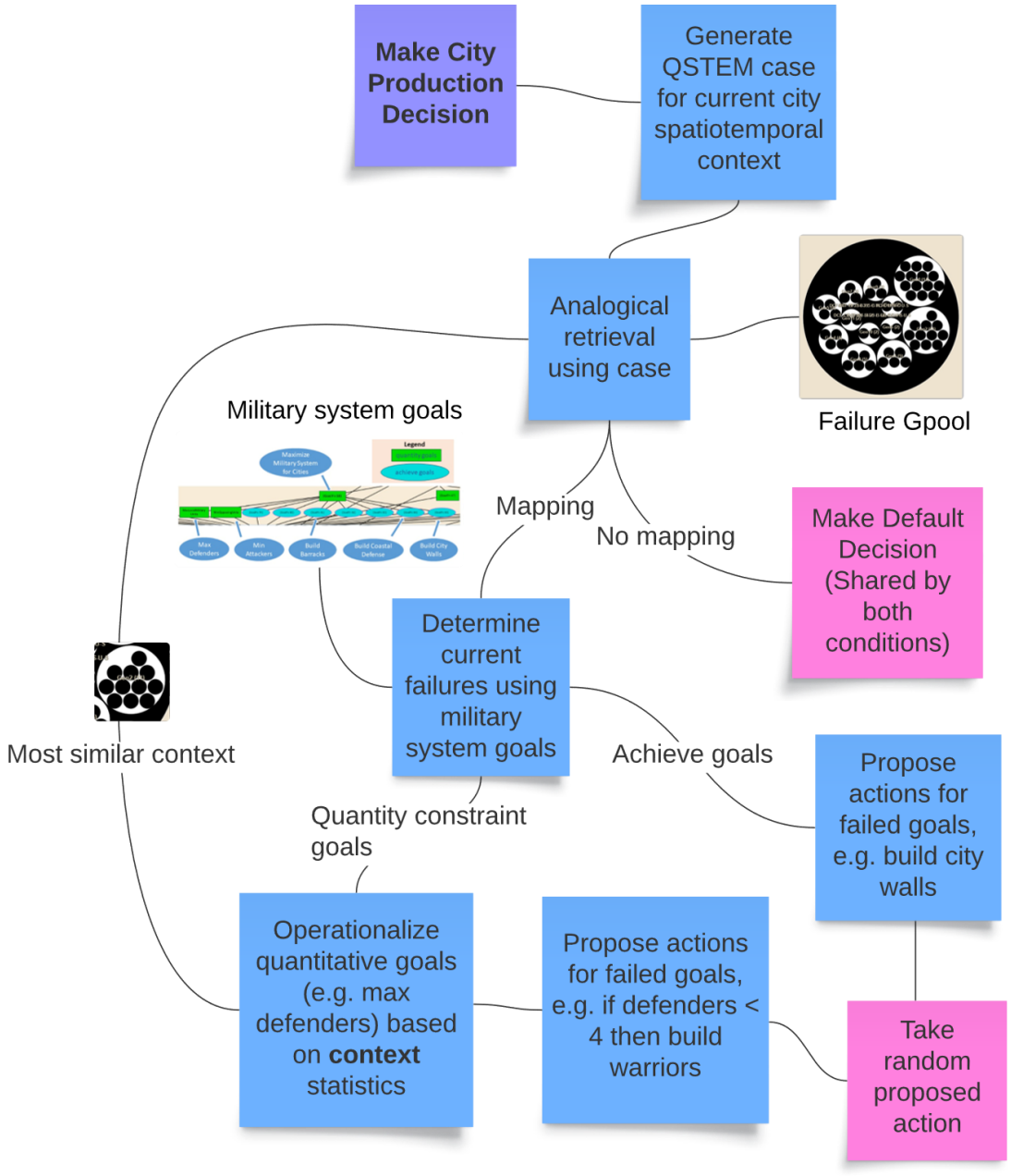


Figure 30: Decision procedure for experiment 2.

sufficiently defended), then a default decision-making procedure[8] is invoked, which is shared across experimental conditions. If the failure gpool is returned, then the next step is to use the goal network to determine an appropriate decision for improving the city's defensive state. Subgoals of the goal to maximize a city's military system (Figure 29) are used for decision making, e.g. building city walls and maximizing the number of military units. First, the agent determines failure for these subgoals. Recall that there are two kinds of goals in the goal network: achieve and quantity constraint goals. Determining failure for achieve goals depends on the current state of the city, e.g. for the goal of having city walls, does the city have them or not? For quantity constraint goals, a learned limit point is used to determine failure. For example, does the city have enough defenders relative to previous losses. To learn limit points, the statistical knowledge that is accumulated in SAGE generalizations is used. There are three main steps:

1. The computed mapping is queried to align current quantities with those from prior experience.
2. Statistics associated with prior quantities are used to determine possible failure.
3. In the case of failure, an action is proposed that would influence the current quantity.

For step one, the analogical mapping is queried to align current quantity values with those from prior experience. Consider the goal to maximize the number of defending military units. The mapping for this quantity might appear like so:

((Defenders (AtFn E1 (StartFn E1))) (UnitCount 1)) →

[8] This procedure was introduced in Hinrichs and Forbus (2016) and leverages the qualitative model to make investment decisions about continuous quantities (e.g. should I build a marketplace to increase gold production). This is the same default procedure as in experiment one.

((Defenders (AtFn (GEFn 0) (StartFn (GEFn 0))) (UnitCount (GEFn 1)).

Here, (GEFn 1) denotes the generalized entity denoting the number of defenders seen at the start of prior battles. Assume that these values are (1 0 1). This means that previously, battle failures have been experienced when there have been one or zero defenders. Step two in the decision-making process is to use these values to determine goal failure depending on the goal direction, i.e. maximization or minimization. For maximization goals, the failure condition is:

(current-quantity ≤ max(previous-quantities)).

For minimization goals, the failure condition is

(current-quantity ≥ min(previous-quantities)).

For example, the current number of defenders is 1, the previously seen quantities are (1 0 1), and this quantity is associated with a maximization goal, so a failure is signaled. The choice of maximum and minimum for failure conditions is simplistic and not robust to outliers. A better discriminator might use statistical information. However, the conditions are the same across experimental conditions, so any suboptimality is normalized. The third and final step is (given a determined failure) to propose an action that would influence the quantity in the desired direction. The agent queries the current execution context for possible actions that the city can take to influence the failed quantity type, e.g. building a new military unit would positively influence the number of defenders. This query depends on existing knowledge from the qualitative model that ties ground actions to quantity goals, which is described in more detail in Hinrichs and Forbus (2016). A set of actions is then proposed for each unsatisfied subgoal. Actions that are not possible are filtered out, e.g. one cannot build a SAM battery before the

scientific advance of rocketry has been discovered. Out of the remaining subset, a decision is chosen at random for the city.

Experiment

There are two conditions for this experiment: baseline and histories. The baseline condition uses nonextended spatial representations, while the histories condition uses footprints. To analyze how event perception affects learning, statistics are given for the size and number of generated cases in each condition. To show that these differences matter for learning, gameplay performance is evaluated using the same metrics as in the first Freeciv experiment, i.e. average city size and gold. Both experimental conditions progress in a similar manner until around turn sixty when enemy invasions begin. The histories condition agent has learned a better model of

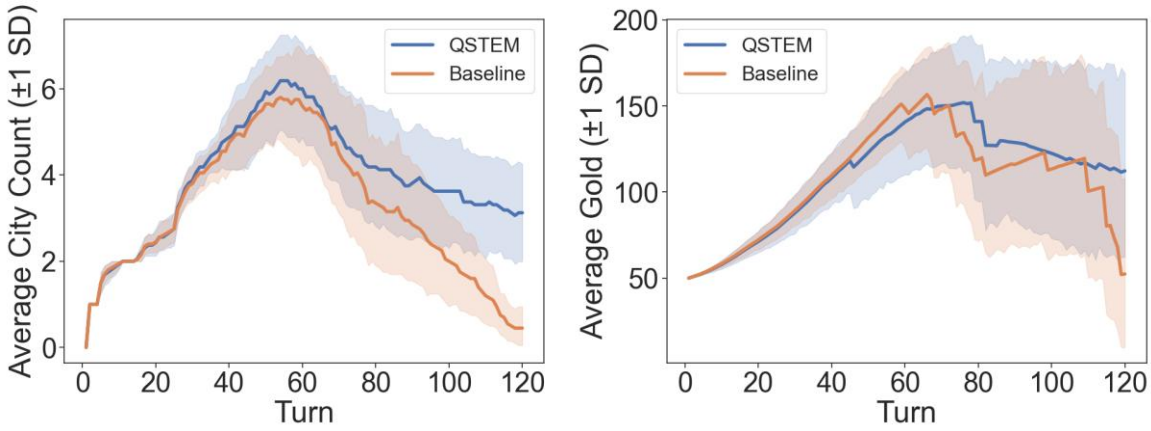


Figure 31: Average number of cities and gold across ten games for baseline and histories conditions.

defense and therefore better defends its cities and maintains its economy (Figure 31). Both results are significant ($p < .05$ using paired t-test).

The number and sizes of cases also varies due to the underlying representations for each condition. For the baseline condition, more episodes are generated, but the average number of facts is fewer. In the histories condition, there are fewer episodes to learn from, but each has more facts on average (Table 2). Even with fewer examples, the histories condition outperforms the baseline. This is despite representational simplifications in Freeciv concerning location for units. Recall that for the baseline condition, location is represented as a point in the Freeciv map. In the game, units can have identical locations, e.g. a player can have multiple archers on the same tile. This means that even without extended space in the baseline condition, it is possible to represent battles with multiple opposing participants. This provides an advantage for the baseline condition which would not hold in the physical world, i.e. physical entities cannot have identical locations.

Condition	Episodes Generated	Average Facts Per Episode
Baseline	171	51.8
Histories	117	69.5

Table 2: Statistics for cases collected from the baseline and histories conditions. The histories condition significantly outperformed the baseline, even though fewer cases were generated. The average number of facts for cases in the histories condition is higher, indicating higher

The learned analogical model makes some simplifying assumptions about the decision space. A better model for military battles would reason about the relative magnitudes of forces explicitly, i.e. a function would be learned. The goal for this experiment is not to learn a perfect model, but to evaluate the role of histories for learning across ceteris paribus modelling assumptions. However, the agent's ability to learn limit points is therefore constrained. The current method limits the agent's ability to generalize, relying on disjunctive distinctions from SAGE to make implicit assumptions about conditions that are relevant for limit points. For example, with two generalizations, one with city walls and the other without, the limit point for

the number of defenders in a city will be different because city walls make defenders more effective. For future work, ordinal relations between different quantity types could be learned based on influences from a causal model. An example from this work is influences on the city military system node, where the agent hypothesizes a limit point between the number of city defenders and enemy attackers. This would allow explicit reasoning about ordinals while generalizing better, which could lead to better decision making.

I have described an experiment in which an agent learns qualitative states of defense in a strategy game from recording spatiotemporal traces of military battles. Empirical evidence from this experiment shows how perception of space affects event individuation and temporal segmentation. I show that history-based representations improve learning because they segment episodes based on change in the environment, and capture knowledge at a useful spatiotemporal grain size. In general, QSTEM can be used as a framework for grounding many kinds of processes in perceptual phenomena. Since histories are useful for representing continuous entities, spilled milk, oil spills, spreading butter are all instances where the kinds of representations used here might help with learning.

CHAPTER 4: CONCEPT LEARNING FOR A COGNITIVE ROBOT

The next set of experiments evaluates the use of QSTEM for concept learning with a robot named AILEEN. Recall that AILEEN learns concepts through interactive experiences with an instructor, who demonstrates events using language and actions. The goal is for the robot to learn language by associating unknown concepts with descriptions of physical change, e.g. the word ‘push’ involves an object moving away from the agent. Previously, AILEEN has learned basic concepts like colors, shapes, and spatial relations (Mohan et al., 2020). One goal of this

work is to extend its episodic learning capabilities to event learning. AILEEN's episodic memory uses the same analogical mechanisms as in the Freeciv experiments, i.e. SME, MAC/FAC, and SAGE.

Event learning remains a difficult task for robotic agents. Many existing multimodal event learners, particularly neural models, map directly from language to fine grained representations, e.g. video. Recent well-documented evidence of adversarial examples indicates that deep learning systems are not actually perceiving and understanding using the kinds of commonsense reasoning that people use, but instead are performing event labeling by exploiting patterns in large data sets that are not generalizable (Liu et al., 2022; Liu et al., 2020). These systems also require vast amounts of training data and energy usage. Event learning poses a particular challenge due to the structured spatiotemporal nature of events. By contrast, I show in this domain that QSTEM facilitates data-efficient incremental learning and the resulting models are inspectable.

Learning in this domain is tested in two experiments: simple and complex event learning. Simple events involve manipulating single objects, e.g. pushing a block, whereas complex events involve change for compound entities, e.g. spreading a group of objects. These experiments show that the same representations and analogical learning stack used for decision making in Freeciv are also useful for concept learning for a robot. QSTEM enables data-efficient learning for AILEEN for the same reasons that it was successful in Freeciv, i.e. the proposed qualitative representations of change enable concise, structured descriptions of events. Additionally, for complex events, reifying compound entities allows for describing aggregate changing properties that are salient for learning, e.g. a spreading event involves the changing volume of a set of objects. Next, I describe how events are represented.

QUALIATIVE COMMONSENSE EVENT LEARNING

This next section describes how AILEEN represents events, including the structure of event descriptions as well as how language is used to guide case construction. Then, the learning and inference processes are described, followed by descriptions of the two experiments and finally a discussion of the results.

EVENT REPRESENTATIONS

Just as in Freeciv, a QSTEM case for AILEEN describes the changing attributes of objects. Here, those objects are parts of the robot (its hand and body), as well as objects in a scene like cubes and cylinders. Figure 32 shows qualitative intervals for a single push event. Recall that qualitative intervals are computed from a series of instantaneous fluent values and an associated temporal encoding (Table 3). For example, the absolute position of AILEEN's hand is associated with the change or quiescence (CQ) encoding, resulting in temporal intervals that describe periods where the hand is moving or stationary. Another example is the forward/backward quantity and associated derivative sign encoding (DS). This describes whether the distance between an object and AILEEN's body is nondecreasing (moving away), nonincreasing (getting closer), or constant. Additionally, RCC8 relations between AILEEN's hand and objects in the scene are reified as intervals. Figure 32 shows three rcc8 intervals (rcc8 hand obj24) which describe a period where the hand is not touching obj24, a period of edge-connectedness, and a period of partial overlap.

Table 3: Quantity definitions and associated encodings. The variable <?obj> is bound to objects in the scene.

Quantity	Encoding
(distance hand <?obj>)	DS
(forwardBackward <?obj>)	DS
(height <?obj>)	DS
(leftRight <?obj>)	DS
(volume <?obj>)	DS
(position hand)	CQ
(position <?obj>)	CQ

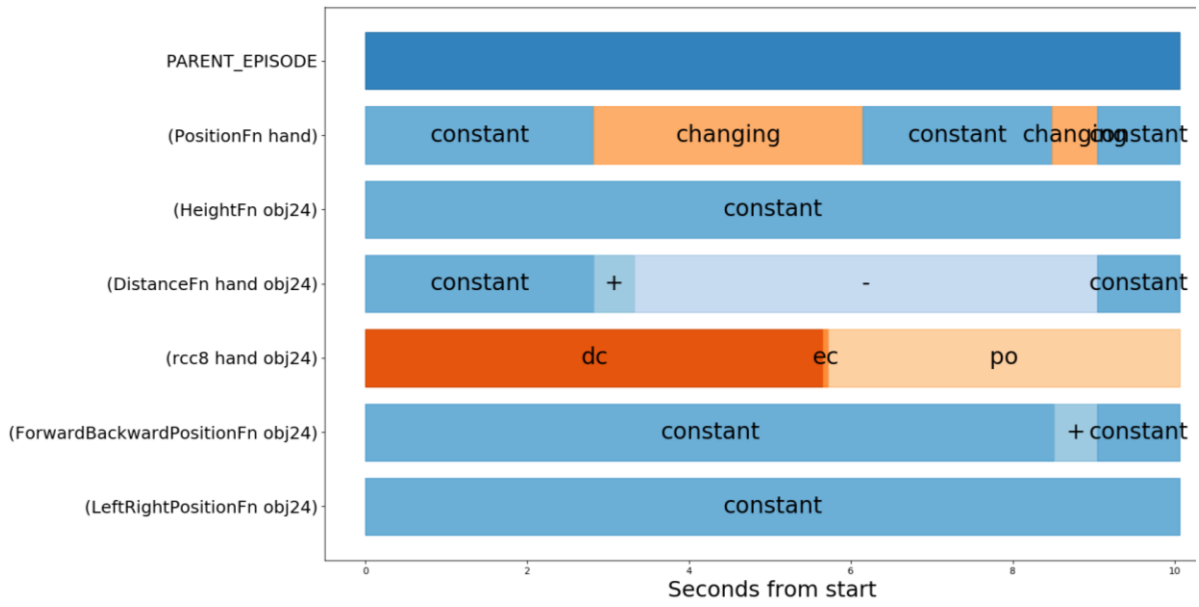


Figure 32: Temporal intervals for a push event.

For the simple event experiments, there is only one object present per demonstration. For the complex event experiment there are multiple objects which are reified as compound entities, and there can be multiple compound entities due to splitting. Figure 33 shows a QSTEM representation for a spread event, which starts with a single compound entity STRegion-c08e and ends with three different compound entities (STRegion-c08e, STRegion-470c, and STRegion-41b4). A history is constructed for each of these entities.

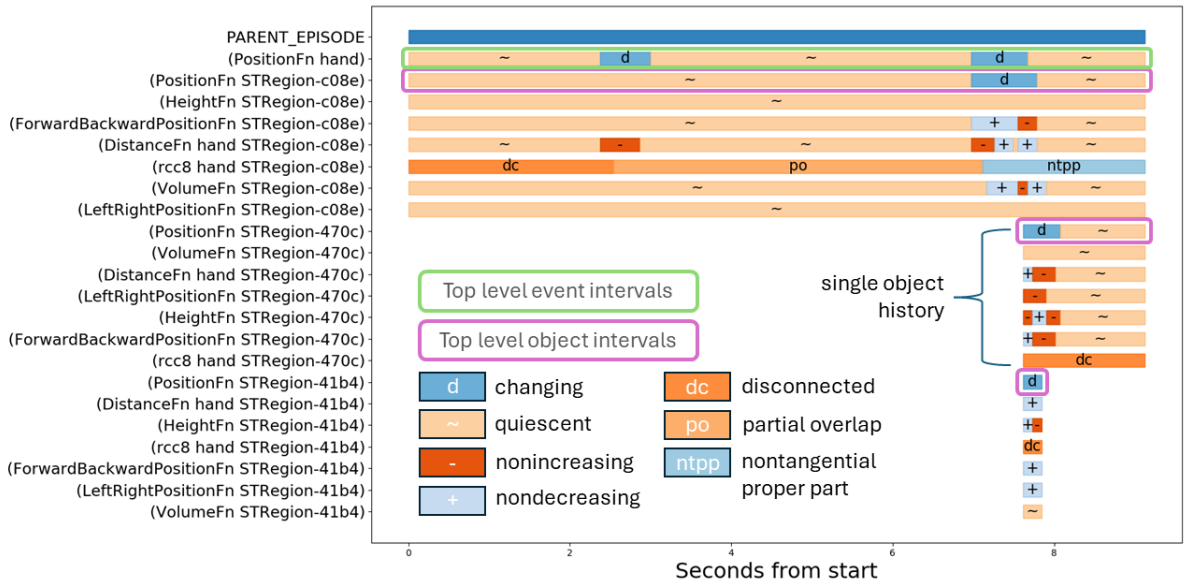


Figure 33: A history for a spread event.

All event descriptions for both experiments use the same event structure, generation of compound entities, definition of persistence, and qualitative encodings as in Freeciv. Here, top-level event intervals are AILEEN's hand movement, i.e. periods where the hand is moving or stationary. Top level object intervals are defined by object movement, again split into periods where each object is moving or stationary. Recall that top-level event intervals are related to top-level object intervals using Allen's Interval Algebra, and top-level object intervals are related to other intervals that describe properties of the object. Table 3 shows the quantities and associated encodings used for this domain. The forwardBackward and leftRight quantities are egocentric from the perspective of the center point of the robot's body. The variable <?obj> is a placeholder that is bound to objects in the scene at a given time. There are two kinds of objects: simple, and compound. Simple objects are single concrete objects like balls and cubes. Compound objects are made up of one or more simple objects. Each simple object's position is defined as the center of its 3D bounding box, and its volume is the volume of that bounding box. The position and

volume of a compound entity are the center of mass[9] and volume of its convex hull, respectively. The kind of object that is reified (i.e. the compound entity or its constituent simple objects) is controlled by linguistic reference, which is described in the next section. RCC8 relations are reified between all referenced objects (simple and complex) and are computed using QSRLib, which takes as input two 3D bounding boxes. A quantization factor of .001 is used.

Grounded Language Learning

Recall that AILEEN learns from an instructor who provides a commentary along with the event demonstration, e.g. 'push the red cube'. The natural language description is used to establish reference to objects in the scene, e.g. red cubes. This is called indexical comprehension (**IC**) (Mohan et al., 2020). Indexical comprehension serves two high level purposes. First, unknown concepts, e.g. the word 'push', trigger the episodic learning process. Second, establishing linguistic reference guides case construction, e.g. a case for the description 'push the red ball' only includes red balls in the scene. AILEEN has previously learned visual concepts (colors and shapes) as well as relations (e.g. left-of) (Mohan et al., 2020), so these concepts are provided as background knowledge. A full description of AILEEN's language-processing mechanism can be found in (Mohan et al., 2020). The general idea is that language is parsed into semantic components, and these components along with background knowledge are used to resolve reference. Background knowledge is represented by bidirectional maps between linguistic concepts, e.g. 'red', and perceptual knowledge, e.g. 'CVRed'. Given a description 'the red cube', the parse returns the semantic components {object-ref1: obj-ref {property:red, property: cube}}. Background knowledge is used to map from linguistic concepts to perceptual knowledge, e.g. the word 'red' is associated with the percept CVRed. Then, the scene is queried,

[9] The system makes an underlying assumption that objects have uniform density

e.g. [and CVRed CVCube] and objects that satisfy the description are returned. When an unknown concept is encountered, e.g. the word ‘push’, the episodic learning process is initiated. The reification of simple vs. compound entities is controlled by reference; simple objects are reified for singular references, e.g. ‘cube’, and compound objects are reified for plurals, e.g. ‘cubes’. With an understanding of how indexical comprehension connects language with perception, the next section describes how QSTEM cases are used for analogical learning.

Event Learning

Recording fluents starts as soon as the environment is initialized, i.e. all objects are generated in the scene, and ends when all objects have come to rest. Quantities are sampled from ground-truth values in the underlying Webots engine. Events are demonstrated on a table in the environment, and only objects that are on the table are reified. That is, in some cases, objects may fall off the table due to the stochastic nature of object placement, i.e. the closer the object is to edge, the more likely it is to fall off the table. This qualitative (on table, <?object>) relation is computed by an object’s centroid being within the three-dimensional prism above the table’s two-dimensional surface. For generating qualitative intervals from fluents, Algorithm 1 is used, where τ is set to .1 seconds and minimum change (mc) is set to .2.

Table 4: SAGE learning parameters for experiments

Parameter	**Experiments 1 and 2**
Assimilation threshold	.7
Probability cutoff	.5
Match threshold	.9

For learning, AILEEN incrementally generalizes event schemas from a series of demonstrations. For each demonstration, a case is generated (hereafter referred to as the probe case) from the recorded trace of the event and the linguistic description, e.g. ‘push’, with a description

"push the green ball". Analogical generalization for AILEEN uses the same processes as in the Freeciv experiments, i.e. a multi-class model is trained and then used for inference. For the Freeciv experiments, a Gpool was trained for each production type, e.g. gold, warriors. The AILEEN experiments use the same setup, where a gpool is trained for each concept, e.g. there is a gpool for *push* and a gpool for *spread* concepts. Recall that training begins with an empty gpool. The first training example is added as an outlier (i.e. an individual case as opposed to a generalization which consists of multiple cases). For the second training example (probe), an analogical retrieval is performed, which returns the item in a gpool that is most similar to the probe case. This retrieval returns a mapping to an item in the gpool (potentially null) along with a similarity score for that mapping. If a mapping is returned and the similarity score is above a fixed assimilation threshold, then the probe is assimilated with the mapped item. If no mapping is returned or the similarity score is below the assimilation threshold, the case is added to the gpool as an outlier. Inference then proceeds as follows. For testing generality, a positive example is generated from an AILEEN demonstration and queried against the gpool corresponding to the positive concept. If a mapping is returned and the corresponding mapping score is equal to or above the match threshold, then AILEEN has correctly classified the case. For testing specificity, a negative example is queried against the gpool for the positive concept. The example is correctly classified if a mapping is not returned or the similarity score is below the match threshold. Parameters used in each experiment are listed in Table 4.

There are two experiments for testing event comprehension. The first experiment examines learning simple events, e.g. pushing and pulling spheres and blocks. The second experiment examines the complex events spreading and partitioning groups of objects, which involve compound entities. The goal for these experiments is for AILEEN to be able to recognize

actions. Learning to perform actions is left for future work, though the learned event schemas might provide a reasonable starting point.

EXPERIMENT ONE: SIMPLE EVENTS

For the first experiment, AILEEN is tested on four kinds of events: push, pull, drop, and pick-up. For each demonstration, a single object is generated and placed on the table. The object's color and shape are chosen randomly from a uniform distribution over a predetermined set of attributes. Objects can have five possible shapes (sphere, cone, block, cylinder, and capsule) and five possible colors (red, green, yellow, blue, and purple). The object is placed at a random position on the table. AILEEN begins each demonstration in the same preset position. A series of movements is defined for each event by specifying a set of vector displacements for AILEEN's hand relative to the target object. Arm movement is computed using the Ikpy[10] inverse kinematics library. For pushing, AILEEN moves its hand towards the object and continues this motion so that the object's motion is away from AILEEN. For pulling, AILEEN extends its arm so that the object is between AILEEN's body and hand, and then moves its hand towards its body, with the same displacement as push. The distinction between pulling and pushing objects has been addressed in qualitative reasoning research (Forbus, 1984). One way to make the distinction is by looking at the direction of applied force; if the force is toward the object, the action is a push, and if the force is away from the object, the action is a pull. Another naive definition, which I use in this work, ignores force and makes a distinction based on egocentric direction of motion, i.e. a push moves an object away from oneself, and a pull moves the object closer. For dropping, AILEEN picks up the object, raises its hand over the object's position, and then ungrasps the object. Pick-up is performed in the same manner as dropping,

[10] https://ikpy.readthedocs.io (Visited on 4/1/2025)

without releasing the object. The idea is that each demonstration, starting from the same initial arm configuration, should result in a single episode for learning, thus simplifying experimentation. It is possible, however, to interpret a dropping demonstration as two subsequent events: first, picking up the object, followed by dropping it. Extending the training regime to more complicated scenarios, e.g. those requiring event segmentation for classifying temporal subintervals as event types, is left for future work.

For the first experiment, 50 cases were generated for each of the four kinds of events, resulting in 200 examples total. Five-fold cross validation was used, where each fold has forty training cases, ten positive test cases, and ten negative test cases. Negative cases are sampled from the set of 150 remaining cases when holding out positive instances, using a uniform distribution. Evaluation uses two metrics: true positives are given by a generality score, and true negatives by a specificity score. Figures for experimental results show average accuracy across five folds given the number of training examples seen. Accuracy is computed as true positives (generality) plus true negatives (specificity) over the number of examples (in all experiments this is ten positive and ten negative examples). The experimental *SAGE* condition is compared against the baseline *FOIL* condition. Both use QSTEM representations; the SAGE condition uses the same analogical learning stack as in the Freeciv experiments, and the FOIL condition uses for First Order Inductive Learner algorithm (Quinlan, 1990). FOIL learns function-free Horn clauses over a set of examples given as background knowledge. It uses information-based covering to incrementally learn a set of rules that increasingly cover subsets of positive and negative examples. Learned rules must match examples exactly, as opposed to analogical matching, which uses a similarity score that measures the degree of structural similarity. For training, the

SAGE condition uses only positive examples, while the FOIL condition uses the same set of positive examples as well as ten held out negative examples.

As the FOIL algorithm learns over function-free Horn clauses, QSTEM representations must be translated because they contain logical functions. This involves a relationalization of logical functions into predicates. For example, for the fact (nondecreasingIn INT1 (HeightFn OBJ2)), (HeightFn OBJ2), which denotes OBJ2's height, must be translated. This is accomplished by reifying the quantity, i.e. (nondecreasingIn INT1 QUANT1) and then relationalizing the logical function, i.e. (height QUANT1 OBJ2). The FOIL condition additionally uses domain dependent constraints on the rules that it can learn, similar to Dubba et al., 2015. In that paper, spatiotemporal rules were learned, and a constraint was added which said that spatial relations, e.g. RCC8 relations must be part of a larger temporal (i.e. Allen relation) structure. For the AILEEN FOIL condition, a similar constraint is used, which says that for any Allen Interval relation, e.g. (aia-meets ?A ?B), either ?A or ?B must appear in a non Allen relation, i.e. they must describe some quantity change or state.

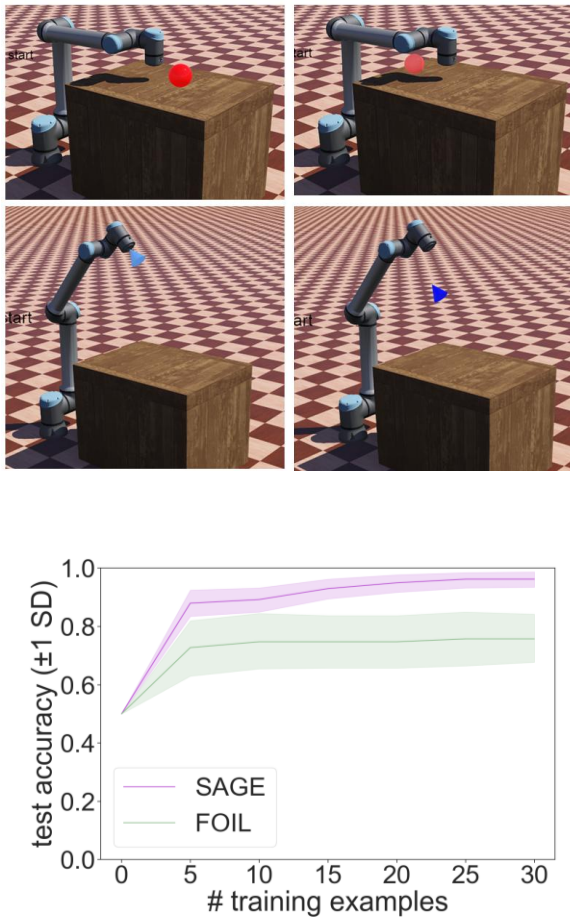


Figure 34: Top: clockwise from top-left: push, pull, pick-up and drop. Bottom: experimental results averaged across all four simple event types.

For this experiment, learning converges before training on the full set of cases, so performance is shown up to thirty training examples (Figure 34). Accuracy at zero examples is shown as 50% for all charts to reflect chance for random binary classification (i.e. a null model that simply predicts the target class for both positive and negative test examples will achieve 50% accuracy). Overall, SAGE significantly outperforms the FOIL condition for simple events ($p<.05$ using paired student's t-test). Results for push show over 95% accuracy with five training examples (Figure 35). For 'pull', accuracy is 100% at 20 examples. The learned event schemas are interpretable, e.g. the learned schema for 'push' contains the facts (during OBJ1-moving-

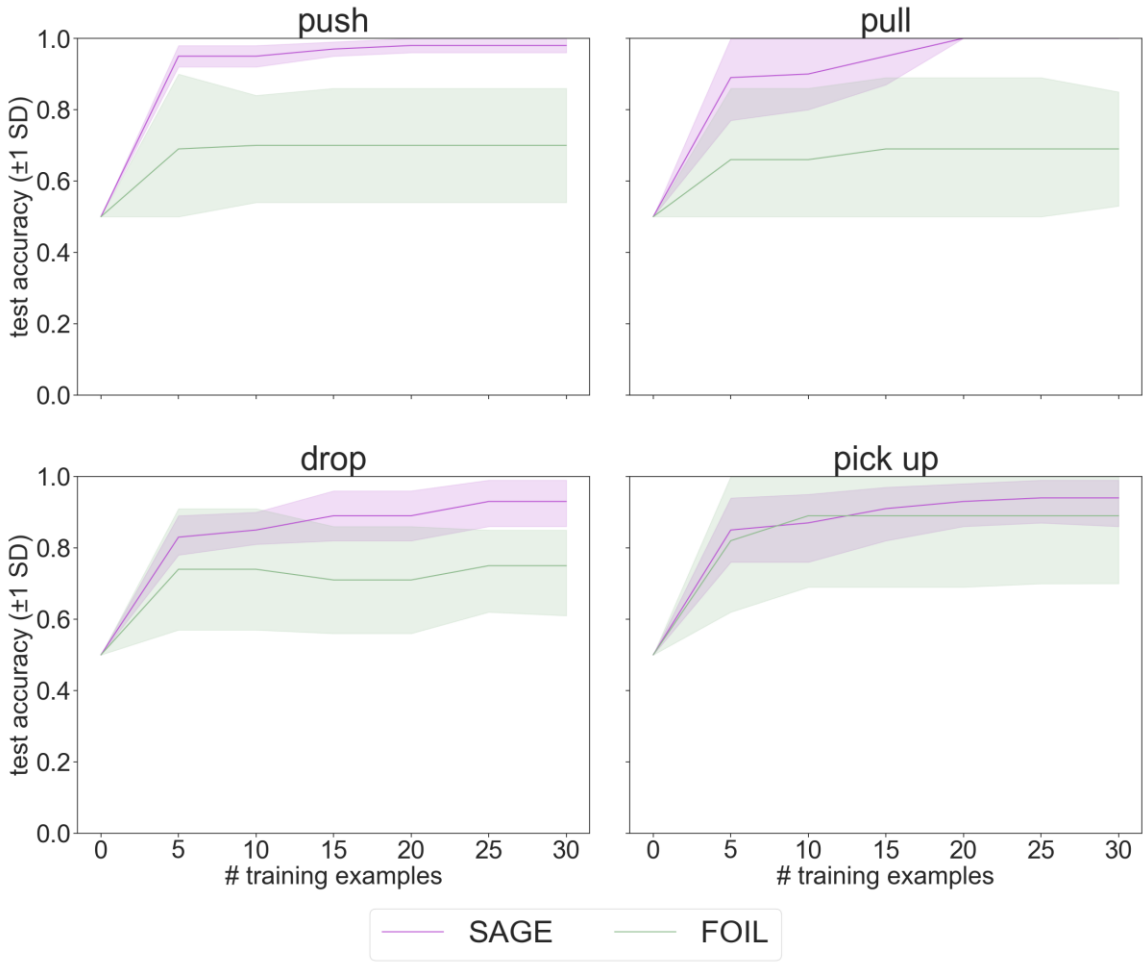


Figure 35: SAGE and FOIL conditions for simple event learning experiments

forward hand-moving) and (meets hand-closer-to-OBJ1 OBJ1-moving), i.e. ‘obj1 is moving away from AILEEN while hand is moving’, and ‘hand is moving closer to the object prior to the object moving’. The learned FOIL rules are also interpretable, but in general do not correspond to intuitive notions of the events being learned. For example, for the first *push* fold, with 30 training examples, FOIL learns the clause (push ?A) ((constantIn ?C ?G) (constantIn ?B ?F) (aia-= ?C ?D) (aia-fi ?C ?B) (aia-f ?B ?A)). This rule effectively states that an interval where some arbitrary quantity is constant, finished by another interval where another arbitrary quantity is constant, implies a push event. Overall, no useful rules were learned by the FOIL condition for simple events, despite relatively high test accuracy.

EXPERIMENT TWO: COMPLEX EVENTS

The second experiment tests AILEEN’s ability to learn complex events which involve multiple objects, i.e. spreading and partitioning. Both cases use the same setup, where a pyramid is built from thirty-six objects. For partitioning, the goal is to divide the pyramid into two main distinct groups of objects. For spreading, the goal is to use AILEEN’s arm to distribute objects on the table while remaining in a single main group. All objects share the same random color and fixed size. Shapes and colors for each object are randomly sampled from the same set as in the simple experiment. For case construction, recall that plural referents in the linguistic description (e.g. ‘the red blocks’) are reified as compound entities. First, indexical comprehension grounds a plural linguistic description in a global set of entities in the scene, e.g. given the signal ‘red blocks’, reference is established to all red blocks. Then, these entities are spatially clustered into 3D regions using the previously described algorithm for computing local footprints. Clustering then computing persistence occurs at each timestep, to describe changes over time, e.g. a group of objects is increasing in volume. The complex behavior of compound entities means that the

resulting cases are much larger than in the simple event experiments, because there are more entities to reify due to object splitting.

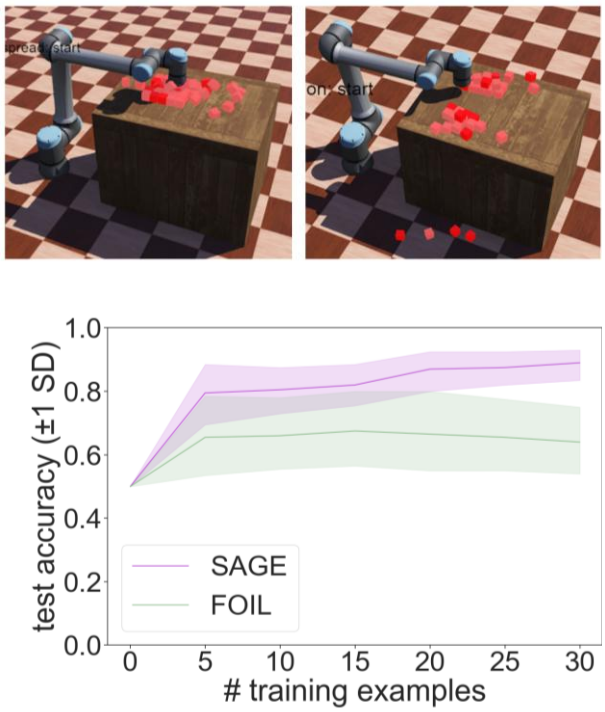


Figure 36: Top: spread and partition demonstrations. Bottom: experimental results averaged across both event types. The baseline condition uses FOIL to learn classification rules inductively.

For experiment two, the SAGE condition significantly outperforms the FOIL baseline condition ($p < .05$ using paired student's t-test) (Figure 36), despite being incremental and not using negative examples for training. In addition to being outperformed by SAGE, the baseline condition does not learn useful aspects of the target events. For example, one learned rule for partition (Figure 37) is (partition A) <= ((changingIn ?C (PositionFn ?G)) (aia-mi ?C ?D) (aia-mi ?C ?B)), which does not encode useful knowledge about the event. On the other hand, the learned analogical models, are inspectable

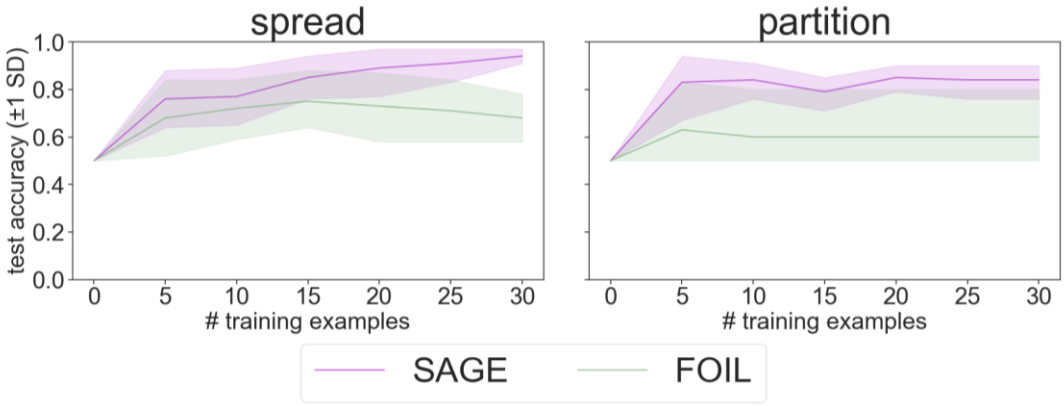


Figure 37: Experimental results for spread and partition complex event learning

and learn useful rules for each event type. For example, the learned model for partition includes an interval in which the volume of an object is decreasing, and the largest spread generalization contains the fact that the volume of one of the compound objects in the scene is increasing over some temporal interval.

Related Work and Discussion

Qualitative representation and reasoning has been applied to both recognition and prediction as well as planning tasks for robotics. Moratz and Tenbrink (2006) built a computational model that used qualitative representations for identifying objects using multiple spatial calculi. They showed that this system was human-like with two empirical human studies. Duckworth et al., (2016) used qualitative spatiotemporal activity graphs (QSTAG) to predict human motion from the perspective of a mobile robot. Do and Pustejovsky (2017) used qualitative spatiotemporal representations for event classification, similar to QSTEM. They used cardinal direction, distance, and trajectory calculi for representation, but did not reify temporal event structure. Qualitative reasoning has also been applied to robotics planning for control problems (Wiley et al., 2016; Wolter and Kirsch, 2015; Wiley et al., 2013), navigation (Boularias et al., 2017; Kuipers and Byun, 1988), and complex task planning (Tenorth and Beetz, 2013), all of which could be future applications for QSTEM.

A particular focus within robotics has been learning from demonstration through situated human-robot interaction. She et al. (2014) studied action learning through human instruction in blocks world. Two planners were used for acting, one continuous, which handled actions like move and grip, and one discrete, which represented synchronous states like on-top-of and on-table. Each planner used distinct representations of the environment; one future direction for QSTEM is to apply the same representation to both continuous control and discrete planning.

One recent focus has been the application of large language models to robot concept and action acquisition (Glocker et al., 2025; Hsieh et al., 2023; Brohan et al., 2023; Song et al., 2023; Ahn et al., 2022). These models often have impressive in-domain performance, but they do not provide insight into how the agents are making decisions and often fail to generalize to new situations or domains (Zhou et al., 2025; Wang et al., 2024), can be vulnerable to bias (Azeem et al., 2024) and introduce safety concerns (Wu et al., 2025).

One possible extension of this work is to use the learned event schemas for task execution, i.e. having the robot learn how to perform the actions. Interactive task learning, or ITL (Laird et al., 2017), describes a framework for situated learning, where knowledge is acquired incrementally over multiple sessions. ITL allows for rich discourse between an instructor and the agent and has previously been used to learn qualitative spatial relations like ‘left-of’ (Mohan et al., 2020). Learning qualitative relations like ‘inside’ or ‘before’ could help an instructor better communicate task requirements to the agent, helping the agent understand how to act in the world.

The representations used in AILEEN focus on regions and RCC relations between them, but there are many more ways to describe space qualitatively that could be useful for robotics, e.g. cardinal directions (Ligozat, G.E., 1998; Frank, A.U., 1992), trajectory calculi (van de Weghe et al., 2006), and distance calculi (Clementini et al., 1997). One potential avenue for future work is to extend QSTEM with these calculi to learn and reason about other kinds of events, e.g. a trajectory calculus may be useful for billiards. One advantage of qualitative representations over purely quantitative ones is that qualitative representations facilitate graceful extension.

For AILEEN, QSTEM uses only qualitative representations for learning, but people also learn from quantitative knowledge. The poverty conjecture (Forbus, 2019) states that there is no general purpose purely qualitative representation of shape or spatial properties. The idea is that people generate descriptions dynamically, based on a given situation. For some tasks, a fine level of detail might be required, and so people use numerical representations. Classic examples are the rolling problem, where an agent must decide if two shapes can roll smoothly against each other (Forbus, 2019) and the CLOCK system (Forbus et al., 1991) which reasoned about moving parts in a clock mechanism. In both, reasoning about shape is crucial to understanding how the system works, e.g. connectivity is determined by fine-grained teeth in the clock system. The metric diagram/place vocabulary model generalizes these ideas by combining qualitative and quantitative representations. The metric diagram represents quantitative aspects of space, e.g. spatial regions with associated quantitative size, and qualitative representations (place vocabularies) are generated from the metric diagram. For event learning in AILEEN, qualitative representations are sufficient for classification, with accuracy being high despite the lack of quantitative knowledge, but these representations would not suffice for more complex tasks, e.g. computing how to grasp an object.

CHAPTER 5: RELATED WORK

QUALITATIVE REPRESENTATION AND REASONING

Histories have been used extensively in qualitative reasoning research, and here I highlight existing work and discuss how this paper improves upon it. Friedman and Forbus (2008) learned qualitative causal models, called protohistories, from simplified natural language descriptions of physical scenarios. A natural language system generated structured cases which

were used for analogical learning of causal models. The resulting models were more akin to situation models, which represent the relationship between episodic memory and natural language, as opposed to this work which utilizes perceptual knowledge. In situation models, dynamic aspects are derived from language properties like tense, as opposed to e.g. motion, so the encoding processes are different. Additionally, that work proposed a method for learning limit points, but the focus was on function-learning, rather than showing how bounded spacetime representations can help with limit point learning as in this work. Friedman and Forbus (2010) looked at learning events like pushing and pulling as a means to study conceptual change. They used CogSketch to generate descriptions of change over time based on comic book frames. The use of comic book frames simplifies event understanding, since the transition from frame to frame assumes some significant change, thus explicitly cueing event segmentation rather than relying on the agent to decide where to segment in a continuous stream of time.

Cohn et al. (2003) proposed using history-based spatiotemporal representations for cognitive vision. They introduced a framework for abducing qualitative descriptions of visual input. It has been used for various types of event and activity recognition tasks, including traffic events (Fernyhough et al., 2000), airport events from video (Sridhar et al., 2010; Dubba et al., 2015), and everyday activities (Alomari et al., 2022; Do and Pustejovsky, 2017). Chen and Forbus (2018) learned action models analogically from skeleton data. These works all share a commonality that they used qualitative spatial representations to aid learning, often using QSRLib (Gatsoulis et al., 2016), a library that computes qualitative spatial representations and calculi from data. While many of these investigations used histories, they all focused solely on events that involve interactions between a limited number of concrete objects. QSTEM extends

this work by reifying complex events as spatiotemporal entities that can have properties, as well as introducing a novel domain-independent representation of qualitative change.

Prior qualitative reasoning work has also looked at formalizing the behavior of regions such as splitting, merging, bursting, and moving, to name a few. Egenhofer and Wilmsen (2006) derived a set of binary topological relations that may hold for a part if a region is split into two parts, which may be useful for reasoning over QSTEM relations. Worboys and Duckham (2006) formalized a framework for modeling continuous spatial change for events and showed how this framework could be used for detecting spatial events such as splitting, merging, hole formation, and hole disappearance. This work shares many similarities with QSTEM, in that it builds qualitative events from snapshot data. However, it is limited to modeling events for single regions. For future work, holes may be relevant to commonsense events not covered by these QSTEM experiments, such as gaps in defensive placements for strategy games, and making donuts for robotic cooking domains.

Learning event models has been studied in language grounding tasks for robotic platforms (Alomari et al., 2022; Alomari et al., 2017; Dubba et al., 2015). Dubba et al. (2015) used qualitative spatiotemporal representations to learn event models from video. The spatial relations used were RCC8 encodings as well as a qualitative trajectory calculus (van de Weghe et al., 2006). QSTEM expands on these representations by reifying bounded events, expanding the types of encodings used, and using analogical learning processes.

Qualitative Spatiotemporal Activity Graphs (QSTAG) (Sridhar et al., 2008; Sridhar et al., 2010; Duckworth et al., 2016; Alomari et al., 2022) are qualitative spatiotemporal representations that share many characteristics with QSTEM. They are at their heart relational

representations that encode qualitative states of entities over time. Using spatial calculi, they can represent concepts like ‘the ball was stationary before it was moving’, or ‘the ball was near the box before the two objects touched’. Alomari et al. (2022) introduced spatiotemporal graphlets; these representations include object attribute information such as color and shape in addition to qualitative spatial change over time in QSTAG. QSTEM builds on these representations in a few ways. For one, the cited spacetime representations tend to be global, relying on the learning algorithm to distinguish signal from noise. On the other hand, QSTEM is biased towards locality, which constrains descriptions considerately. Additionally, QSTEM introduces new representations of continuous change in addition to change/no change representations in QSTAG. Finally, QSTEM generalizes event representations by describing compound entities, which enables reasoning about persistence for military battles and complex events like spreading objects.

WORK IN GAME AI

One major focus in the cognitive game AI community has been learning from experience. Weber and Mateas (2009a; 2009b) used Conceptual Neighborhoods (Freksa and Kreutzmann, 2017; Freksa, 1992) for learning build order in a real-time strategy game. Case-based retrieval and adaptation operated over a latent conceptual neighborhood space, whereas QSTEM uses a form of conceptual neighborhoods in the spatial sense (reifying RCC relations between entities as fluents in event descriptions). Both Weber and Mateas as well as the QSTEM Freeciv experiments use production decision optimization for the experimental task, but the former used a production type specific and hand-coded retrieval mechanism, whereas QSTEM and analogical retrieval are domain-general. Furthermore, their system did not represent events or model episodic memory. Other examples of game AI applications from the case-based learning

community include work on goal-driven autonomy (Klenk et al., 2013; Weber et al., 2010; Molineaux et al., 2010; Aha et al., 2005), where the focus is on planning and goal creation and maintenance under uncertainty. These works may provide inspiration for future work using QSTEM for planning and reasoning under uncertainty.

There is also a history of using episodic memory in strategy game domains. Work on episodic memory in the SOAR cognitive architecture has been applied to a simulated military domain (Nuxoll and Laird, 2007) called TankSOAR. The agent learned tactics by evaluating past experience; memories were represented as localized world states at the instantaneous time of sensing, so events were not temporally extended and therefore unstructured. TankSoar is much less complex than Freeciv, with only tanks in a smaller map, and with about ten possible actions. Another use of episodic memory in game AI has been narrative understanding and social reasoning. Forbus and Kuehne (2007) studied how episodic memory could enhance interactions with non-playable characters (NPCs), as did (Brom et al., 2007), but neither used histories.

Spatial and temporal representations and reasoning have also been used in strategy games. Madeira et al. (2006) used spatial abstractions of a game map (i.e. regions useful for decision making) to reduce the sparsity of rewards for reinforcement learning, though no temporal representations were used. The study of how spatial and temporal reasoning can be used for decision making has been addressed in a tower defense video game (Wetzel et al., 2012), where a think aloud protocol was used to record strategies of human players. Three strategies were identified, two temporal and one spatial. The underlying assumption of the authors was that the top-level goal of players was to optimize a temporal problem (maximizing the amount of time that the player's towers could attack the invading hordes), but that players often recast this problem in terms of spatial properties, as determined by their verbal descriptions

of their strategies. The idea was that temporal reasoning could be rerepresented as spatial reasoning, and players that used the spatial strategy outperformed the purely temporal strategy. Both that work as well as QSTEM share the goal of demonstrating how spatial thinking can organize perception of time.

EPISODIC MEMORY AND MACHINE LEARNING

While episodic memory has been studied in AI for quite some time (Nuxoll and Laird, 2004; Forbus and Kuehne, 2007; Anderson et al., 1998; McClelland et al., 1995; Schank, 1980), recent developments in large language models have prompted an effort to understand how long-term memory can be useful for neural AI systems. Due to the nature of their training regime, LLMs often suffer from forgetting, i.e. they have no situated context. The current solution is to augment prompts with relevant prior knowledge of situation. This suffers from two major problems; 1: it is unclear what the functional limits on context length are, and 2: context management falls outside of the scope of the LLM itself and must be managed by other systems. An additional problem is the lack of sample efficiency at train time. As a result, there has been a growing effort to build neural systems that explicitly store and retrieve previous memories, including architectures that are biased towards spatiotemporal information (Parisi et al., 2018; Chang and Tan, 2017). With these architectures come problems associated with neural models, including data inefficiency, limits on reasoning capabilities, and limitations on interpretability and underlying representational structures. For the latter, it is currently unclear how known structures such as hierarchy, partonomy, and Davidsonian event structure could be encoded and learned in a neural substrate in the context of stochastic gradient descent-based training methods. Huet et al., 2025 proposed a spatiotemporal architecture for episodic memory, but is limited to

synchronic events. Freire et al. (2025) used episodic memory to improve a deep reinforcement learner, but temporal episode duration was fixed and event structure was not explicitly encoded.

There have also been attempts to formalize the realization of event structure within the substrate of a neural architecture (Healy and Caudell, 2019). There, the goal is to build neural models that realize event structure, including conceptual hierarchy and the grounding of participant structure. This theoretical work could provide insight into how a conceptual framework such as QSTEM could be realized in the brain, but it is unclear how the model could be trained.

CHAPTER 6: DISCUSSION AND FUTURE WORK

This thesis presents a novel event representation that uses qualitative history-based representations to describe events. These representations describe local, bounded pieces of spacetime and change within them. Descriptions of change use novel domain general qualitative encodings of quantity and reify change as temporal intervals. I also introduce a novel qualitative spatial representation called a footprint, which reifies compound spatial entities as first-class entities. These compound representations serve two main functions: they allow perceptual knowledge to constrain case construction to knowledge that is salient for a given situation, and they form the basis for construing many kinds of events as histories, providing an event structure that uses Davidsonian attributes for qualitative properties. These representations allow for interpretable and data efficient event learning, and I show empirical evidence for this in two domains: a complex strategy game and a cognitive robotics environment.

The analogical learning mechanisms used for learning have been applied to many different domains from sketch understanding (Chen et al., 2023), theory of mind (Rabkina et al.,

2018), question answering (Crouse et al., 2018), and natural language debugging (Nakos et al., 2020) and so I hypothesize that it will be further useful for investigating cognitive models of episodic memory. SAGE has scaled to thousands of examples, with hundreds of facts per example (Chen and Forbus, 2021), and so I hypothesize that it will scale to even more complex domains for studying episodic memory.

LIMITATIONS

Spatial and temporal representation in humans is complex. The focus of QSTEM is on the role of spatial and temporal regions and their role in organizing memory, but there are more ways to represent space. Many more qualitative spatial and spatiotemporal calculi have been described (see Chen et al., 2015), e.g. cardinal directions (Ligozat, G.E., 1998; Frank, A.U., 1992), trajectory calculi (van de Weghe et al., 2006), distance calculi (Clementini et al., 1997), and orientation calculi (Freksa, 1992). QSTEM then provides a representational mechanism that can be gracefully extended with these kinds of descriptions. Spatiotemporal locality heuristics will also constrain any additional knowledge about entities, so the hypothesis is that the cases will still be data-efficient, all else being equal.

People also use metric representations when thinking about space. For example, grid cells (Moser et al., 2014) encode knowledge about distance, direction, and location. In qualitative reasoning research, place vocabularies (Forbus, 2019) provide a formalism for unifying coordinate and qualitative representations of space. One potential direction for future work is to study how these schemas can be used for episodic memory.

REASONING IN NOISY ENVIRONMENTS

Both Freeciv as well as Webots are simulated environments and do not explicitly introduce noise into sampling the respective environments[11]. A question then arises regarding the applicability of QSTEM to real world scenarios. In general, it has been argued that qualitative representation and reasoning is useful in noisy environments, in part because qualitative abstraction improves robustness by tolerating errors in metric input data, e.g. GPS jitter (Cohn and Renz, 2008). This has been demonstrated empirically when using qualitative representations for learning events from video (Dubba et al., 2015) and spatial scene understanding (Krishnaswamy et al., 2018).

EVENT PERCEPTION

QSTEM presents a model of event segmentation where knowledge about when to segment an event is provided as background knowledge. In any reasonably complex scenario, there are infinite ways to carve up time. The knowledge that has been provided to the agent has been hand coded; future research may investigate how this knowledge is acquired. Event perception literature has found that certain kinds of knowledge (e.g. goals and causal information) structure the temporal aspect of events; future work may investigate how process models could be used to automatically carve up time based on reasoning, for example event boundaries may correspond with quantitative limit points. A simple example is thinking about events for boiling water, where time is segmented into a period of increasing temperature followed by a period where the water is boiling. In Freeciv experiment two, the learned causal

[11] In AILEEN, quantities sampled from the underlying engine (e.g. object position) must be smoothed due to minute fluctuations likely due to numerical precision.

model is used for learning within event contexts, but could possibly be used to influence event perception.

Each generated case in both domains represents a single event, but future work may investigate the application of QSTEM to simultaneous or overlapping events, e.g. two robot arms working together. A simple way to implement this within the existing QSTEM framework would be to introduce additional top-level event interval types, and reify temporal relations between these intervals, e.g. the robot's left arm is moving at the same time as the right arm.

GENERALISABILITY AND FUTURE WORK

I see QSTEM as a general framework for representing structured, hierarchical aspects of spatiotemporal events. The use of histories situates the underlying representation in a long history of commonsense representation and reasoning. The mixture of continuous and discrete representations of change enable describing many kinds of phenomena. Representations of compound entities simplify descriptions as well as provide a useful abstraction for semantic grounding, e.g. describing how the concept 'spread' can be described by the compound entity changing in size. In this sense, QSTEM can be applied to many situations. Groups of military units, flocks, liquids, etc. are examples of compound entities, as well as many concrete entities that have parts, e.g. organisms, cities, and machines. Often they are stable in their configuration, but fluid commonsense reasoning, particularly with respect to persistence, requires reasoning about changing parts. Tibbles the cat losing a tail remains Tibbles, and I am still me when I lose a tooth. Conceptually, named entities are inherently stable, but we often make judgements about how these entities persist based on visual perceptual knowledge. This thesis shows how histories are useful for this kind of reasoning. The key is the spatiotemporal continuity is required before making judgements on persistence. Reasoning about persistence is not only useful for compound

entities, but events in general. The second Freeciv experiment shows how qualitative spatial representations can be used for reasoning about the persistence of military battles. Again, the emphasis here is how spatial representations can be leveraged to determine when events start and end. The spatially bounded nature of histories is key to this kind of reasoning.

I see applications for QSTEM in many other AI subdomains. Domain general representations for episodic memory may prove useful for social reasoning. From the perspective of representation, many aspects of social situations can be construed as processes, e.g. classes of social phenomena that influence trust. Qualitative representations can then be used for analogical episodic learning and reasoning. Another potential use is for interactive task learning (Laird et al., 2017), where humans collaborate with AI systems to teach them how to achieve complex goals. Spatiotemporal representations may help ground collaborative problem solving in the same way that they help with language learning.

QSTEM is cognitively inspired and helps with human-like decision making in a strategy game. Another research direction is investigating its ability to model cognition. Building cognitive models of event perception could further help learning and reasoning. There is a rich body of work on Event Perception involving human subjects that could be leveraged to this end.

CONCLUSION

I have introduced a novel qualitative spatiotemporal event representation called QSTEM and showed how it can support episodic learning in two domains: a complex strategy game and a cognitive robot. QSTEM is built on Pat Hayes' notion of histories, and is inspired by psychological models of event perception, which are used to generate histories. I show that QSTEM is useful for learning, in part because histories are useful for reasoning about event semantics, because they support reasoning about persistence. QSTEM is domain general, where

the same event structure, compound representations, descriptions of change, definition of persistence, and analogical learning mechanism are used for the strategy game and the robotics work.

REFERENCES

Aha, D. W., Molineaux, M., & Ponsen, M. (2005). Learning to Win: Case-Based Plan Selection in a Real-Time Strategy Game. In H. Muñoz-Ávila & F. Ricci (Eds.), *Case-Based Reasoning Research and Development* (Vol. 3620, pp. 5–20). Springer Berlin Heidelberg. https://doi.org/10.1007/11536406_4

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Fu, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Ho, D., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jang, E., Ruano, R. J., Jeffrey, K., … Zeng, A. (2022). *Do As I Can, Not As I Say: Grounding Language in Robotic Affordances* (arXiv:2204.01691). arXiv. https://doi.org/10.48550/arXiv.2204.01691

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, *26*(11), 832–843. https://doi.org/10.1145/182.358434

Alomari, M., Duckworth, P., Hogg, D., & Cohn, A. (2017). Natural Language Acquisition and Grounding for Embodied Robotic Systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, *31*(1). https://doi.org/10.1609/aaai.v31i1.11161

Alomari, M., Li, F., Hogg, D. C., & Cohn, A. G. (2022). Online perceptual learning and natural language acquisition for autonomous robots. *Artificial Intelligence*, *303*, 103637. https://doi.org/10.1016/j.artint.2021.103637

Anderson, J. R., Bothell, D., Lebiere, C., & Matessa, M. (1998). An Integrated Theory of List Memory. *Journal of Memory and Language*, *38*(4), 341–380. https://doi.org/10.1006/jmla.1997.2553

Azeem, R., Hundt, A., Mansouri, M., & Brandão, M. (2024). *LLM-Driven Robots Risk Enacting Discrimination, Violence, and Unlawful Actions* (arXiv:2406.08824). arXiv. https://doi.org/10.48550/arXiv.2406.08824

Boularias, A., Duvallet, F., Oh, J., & Stentz, A. (2017). *Learning Qualitative Spatial Relations for Robotic Navigation*.

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. https://doi.org/10.1613/jair.3484

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., Ding, T., Driess, D., Dubey, A., Finn, C., Florence, P., Fu, C., Arenas, M. G., Gopalakrishnan, K., Han, K., Hausman, K., Herzog, A., Hsu, J., Ichter, B., … Zitkovich, B. (2023). *RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control* (arXiv:2307.15818). arXiv. https://doi.org/10.48550/arXiv.2307.15818

Brom, C., Pešková, K., & Lukavský, J. (2007). What Does Your Actor Remember? Towards Characters with a Full Episodic Memory. In M. Cavazza & S. Donikian (Eds.), *Virtual Storytelling. Using Virtual Reality Technologies for Storytelling* (pp. 89–101). Springer. https://doi.org/10.1007/978-3-540-77039-8_8

Chang, P.-H., & Tan, A.-H. (2017). Encoding and Recall of Spatio-Temporal Episodic Memory in Real Time. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 1490–1496. https://doi.org/10.24963/ijcai.2017/206

Chen, J., Cohn, A. G., Liu, D., Wang, S., Ouyang, J., & Yu, Q. (2015). A Survey of Qualitative Spatial Representations. *The Knowledge Engineering Review*, *30*(1), 106–136. https://doi.org/10.1017/S0269888913000350

Chen, K., & Forbus, K. (2018). Action Recognition From Skeleton Data via Analogical Generalization Over Qualitative Representations. *Proceedings of the AAAI Conference on Artificial Intelligence*, *32*(1), Article 1. https://doi.org/10.1609/aaai.v32i1.11328

Chen, K., Rabkina, I., McLure, M. D., & Forbus, K. D. (2019). Human-Like Sketch Object Recognition via Analogical Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(01), Article 01. https://doi.org/10.1609/aaai.v33i01.33011336

Chen, K., & Forbus, K. (2021). *Visual Relation Detection Using Hybrid Analogical Learning*.

Chen, K., Forbus, K., Srinivasan, B. V., & Chhaya, N. (2023). *Sketch Recognition via Part-based Hierarchical Analogical Learning*.

Clementini, E., Felice, P. D., & Hernández, D. (1997). Qualitative representation of positional information. *Artificial Intelligence*, *95*(2), 317–356. https://doi.org/10.1016/S0004-3702(97)00046-5

Cohn, A. G., Bennett, B., Gooday, J., & Gotts, N. M. (1997). Qualitative Spatial Representation and Reasoning with the Region Connection Calculus. *GeoInformatica*, *1*(3), 275–316.

Cohn, A. G., Magee, D. R., Galata, A., Hogg, D. C., & Hazarika, S. M. (2003). Towards an Architecture for Cognitive Vision Using Qualitative Spatio-temporal Representations and Abduction. In C. Freksa, W. Brauer, C. Habel, & K. F. Wender (Eds.), *Spatial Cognition*

III (Vol. 2685, pp. 232–248). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-45004-1_14

Cohn, A. G., & Renz, J. (2008). Qualitative Spatial Representation and Reasoning. In F. van Harmelen, V. Lifschitz, & B. Porter (Eds.), *Handbook of Knowledge Representation* (Vol. 3, pp. 551–596). Elsevier. https://doi.org/10.1016/S1574-6526(07)03013-1

Cohn, A., & Gotts, N. (1995). *The `Egg-Yolk' Representation Of Regions with Indeterminate Boundaries*. *2*.

Crouse, M., McFate, C., & Forbus, K. (2018). *Learning from Unannotated QA Pairs to Analogically Disambiguate and Answer Questions*. 9.

Do, T., & Pustejovsky, J. (2017, October 2). *Learning event representation: As sparse as possible, but not sparser*. Qualitative Reasoning Workshop 2017. https://arxiv.org/abs/1710.00448v1

Dubba, K. S. R., Cohn, A. G., Hogg, D. C., Bhatt, M., & Dylla, F. (2015). Learning Relational Event Models from Video. *Journal of Artificial Intelligence Research*, *53*, 41–90. https://doi.org/10.1613/jair.4395

Duckworth, P., Gatsoulis, Y., Jovan, F., Hawes, N., Hogg, D. C., & Cohn, A. G. (2016). Unsupervised Learning of Qualitative Motion Behaviours by a Mobile Robot. *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, 1043–1051.

Egenhofer, M. J., & Wilmsen, D. (2006). Changes in Topological Relations when Splitting and Merging Regions. In A. Riedl, W. Kainz, & G. A. Elmes (Eds.), *Progress in Spatial Data*

Handling: 12th International Symposium on Spatial Data Handling (pp. 339–352). Springer. https://doi.org/10.1007/3-540-35589-8_22

Fernyhough, J., Cohn, A. G., & Hogg, D. C. (2000). Constructing qualitative event models automatically from video input. *Image and Vision Computing*, *18*(2), 81–103. https://doi.org/10.1016/S0262-8856(99)00023-2

Forbus, K. D. (1984). Qualitative Process Theory. *Artificial Intelligence*, 85–168.

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

Forbus, K. D., Ferguson, R. W., Lovett, A., & Gentner, D. (2017). Extending SME to Handle Large-Scale Cognitive Modeling. *Cognitive Science*, *41*(5), 1152–1201.

Forbus, K. D., Gentner, D., & Law, K. (1995). MAC/FAC: A model of similarity-based retrieval. *Cognitive Science*, *19*(2), 141–205. https://doi.org/10.1016/0364-0213(95)90016-0

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

Forbus, K. D., & Hinrichs, T. R. (2006). *Companion Cognitive Systems: A step towards human-level AI*. 5.

Forbus, K. D., & Kuehne, S. (2007). *Episodic Memory: A Final Frontier*. 4.

Forbus, K. D., Nielsen, P., & Faltings, B. (1991). Qualitative spatial reasoning: The CLOCK project. *Artificial Intelligence*, *51*(1–3), 417–471. https://doi.org/10.1016/0004-3702(91)90116-2

Forbus, K., Usher, J., Lovett, A., Lockwood, K., & Wetzel, J. (2011). CogSketch: Sketch Understanding for Cognitive Science Research and for Education: Topics in Cognitive Science. *Topics in Cognitive Science*, *3*(4), 648–666.

Frank, A. U. (1992). Qualitative spatial reasoning about distances and directions in geographic space. *Journal of Visual Languages & Computing*, *3*(4), 343–371. https://doi.org/10.1016/1045-926X(92)90007-9

Freire, I. T., Amil, A. F., & Verschure, P. F. M. J. (2025). Sequential memory improves sample and memory efficiency in episodic control. *Nature Machine Intelligence*, *7*(1), 43–55. https://doi.org/10.1038/s42256-024-00950-3

Freksa, C. (1992). Using Orientation Information for Qualitative Spatial Reasoning. *Proceedings of the International Conference GIS - From Space to Territory: Theories and Methods of Spatio-Temporal Reasoning on Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, 162–178.

Freksa, C., & Kreutzmann, A. (2017). Conceptual Neighborhoods. In D. Richardson, N. Castree, M. F. Goodchild, A. Kobayashi, W. Liu, & R. A. Marston (Eds.), *International Encyclopedia of Geography: People, the Earth, Environment and Technology* (pp. 1–12). John Wiley & Sons, Ltd. https://doi.org/10.1002/9781118786352.wbieg0935

Friedman, S. E., & Forbus, K. D. (2008). Learning Qualitative Causal Models via Generalization & Quantity Analysis. *Proceedings of the 22nd International Workshop on Qualitative Reasoning*.

Friedman, S., & Forbus, K. (2010). An Integrated Systems Approach to Explanation-Based Conceptual Change. *Proceedings of the AAAI Conference on Artificial Intelligence*, *24*(1), 1523–1529. https://doi.org/10.1609/aaai.v24i1.7572

Gatsoulis, Y., Alomari, M., Burbridge, C., Dondrup, C., Duckworth, P., Lightbody, P., Hanheide, M., Hawes, N., Hogg, D. C., & Cohn, A. G. (2016). QSRlib: A software library for online acquisition of Qualitative Spatial Relations from Video. *29th International Workshop on Qualitative Reasoning*.

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

Glocker, M., Hönig, P., Hirschmanner, M., & Vincze, M. (2025). *LLM-Empowered Embodied Agent for Memory-Augmented Task Planning in Household Robotics* (arXiv:2504.21716). arXiv. https://doi.org/10.48550/arXiv.2504.21716

Goel, A. K., & Jones, J. (2011). Metareasoning for Self-Adaptation in Intelligent Agents. *Metareasoning: Thinking about Thinking*, 151.

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

Hayes, P. J. (1979). The Naive Physics Manifesto. *Expert Systems in the Microelectronic Age*. https://cir.nii.ac.jp/crid/1570009749139377408

Hayes, P. J. (1989). Naive Physics I: Ontology for Liquids. In *Readings in qualitative reasoning about physical systems* (pp. 484–502). Morgan Kaufmann Publishers Inc.

Hayes, P. J. (1995). *The Second Naive Physics Manifesto*. https://doi.org/10.1016/B978-1-4832-1447-4.50010-9

Hazarika, S. M., & Cohn, A. G. (2001). Qualitative Spatio-Temporal Continuity. In D. R. Montello (Ed.), *Spatial Information Theory* (pp. 92–107). Springer. https://doi.org/10.1007/3-540-45424-1_7

Hazarika, S. M., & Cohn, A. G. (2002). *Abducing Qualitative Spatio-Temporal Histories from Partial Observations*.

Healy, M. J., & Caudell, T. P. (2019). Episodic memory: A hierarchy of spatiotemporal concepts. *Neural Networks*, *120*, 40–57. https://doi.org/10.1016/j.neunet.2019.09.021

Hinrichs, T., & Forbus, K. (2012). Learning Qualitative Models by Demonstration. *Proceedings of the AAAI Conference on Artificial Intelligence*, *26*, Article 1. https://doi.org/10.1609/aaai.v26i1.8153

Hinrichs, T. R., & Forbus, K. D. (2016). Qualitative Models for Strategic Planning. *Proceedings of the Third Annual Conference on Advances in Cognitive Systems*, 18.

Hsieh, C.-Y., Chen, S.-A., Li, C.-L., Fujii, Y., Ratner, A., Lee, C.-Y., Krishna, R., & Pfister, T. (2023). *Tool Documentation Enables Zero-Shot Tool-Usage with Large Language Models* (arXiv:2308.00675). arXiv. https://doi.org/10.48550/arXiv.2308.00675

Huet, A., Houidi, Z. B., & Rossi, D. (2025). *Episodic Memories Generation and Evaluation Benchmark for Large Language Models* (arXiv:2501.13121). arXiv. https://doi.org/10.48550/arXiv.2501.13121

Kandaswamy, S., & Forbus, K. D. (2012). Modeling Learning of Relational Abstractions via Structural Alignment. *Proceedings of the 34th Annual Meeting of the Cognitive Science Society*, 6.

Klenk, M., Molineaux, M., & Aha, D. W. (2013). Goal-Driven Autonomy for Responding to Unexpected Events in Strategy Simulations. *Computational Intelligence*, *29*(2), 187–206. https://doi.org/10.1111/j.1467-8640.2012.00445.x

Krishnaswamy, N., Friedman, S., & Pustejovsky, J. (2018). *Combining Deep Learning and Qualitative Spatial Reasoning to Learn Complex Structures from Sparse Examples with Noise* (arXiv:1811.11064). arXiv. https://doi.org/10.48550/arXiv.1811.11064

Kuipers, B. J., & Byun, Y.-T. (1988). A robust, qualitative method for robot spatial learning. *Proceedings of the Seventh AAAI National Conference on Artificial Intelligence*, 774–779.

Laird, J. E. (2019). *The Soar Cognitive Architecture*. MIT Press.

Laird, J. E., & Derbinsky, N. (2009). *A Year of Episodic Memory*. Workshop on Grand Challenges for Reasoning from Experiences.

Laird, J. E., Gluck, K., Anderson, J., Forbus, K. D., Jenkins, O. C., Lebiere, C., Salvucci, D., Scheutz, M., Thomaz, A., Trafton, G., Wray, R. E., Mohan, S., & Kirk, J. R. (2017). Interactive Task Learning. *IEEE Intelligent Systems*, *32*(4), 6–21. https://doi.org/10.1109/MIS.2017.3121552

Lenat, D. B. (1995). CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, *38*(11), 33–38. https://doi.org/10.1145/219717.219745

Ligozat, G. É. (1998). Reasoning about Cardinal Directions. *Journal of Visual Languages & Computing*, *9*(1), 23–44. https://doi.org/10.1006/jvlc.1997.9999

Liu, A., Huang, T., Liu, X., Xu, Y., Ma, Y., Chen, X., Maybank, S. J., & Tao, D. (2020). *Spatiotemporal Attacks for Embodied Agents* (arXiv:2005.09161). arXiv. https://doi.org/10.48550/arXiv.2005.09161

Liu, F., Liu, H., & Jiang, W. (2022). *Practical Adversarial Attacks on Spatiotemporal Traffic Forecasting Models* (arXiv:2210.02447). arXiv. https://doi.org/10.48550/arXiv.2210.02447

Lovett, A., & Forbus, K. (2017). Modeling visual problem solving as analogical reasoning. *Psychological Review*, *124*(1), 60–90.

Madeira, C., Corruble, V., & Ramalho, G. (2006). Designing a Reinforcement Learning-Based Adaptive AI for Large-Scale Strategy Games. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, *2*(1), Article 1. https://doi.org/10.1609/aiide.v2i1.18759

McClelland, J. L., McNaughton, B. L., & O'Reilly, R. C. (1995). Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, *102*(3), 419–457. https://doi.org/10.1037/0033-295X.102.3.419

McLure, M., & Forbus, K. (2012). Encoding Strategies for Learning Geographical Concepts via Analogy. *Proceedings of the Twenty-Sixth International Workshop on Qualitative Reasoning*.

Menager, D. H., & Choi, D. (2016). A Robust Implementation of Episodic Memory for a Cognitive Architecture. *38th Annual Meeting of the Cognitive Science Society*.

Michel, O. (2004). WebotsTM: Professional Mobile Robot Simulation. *International Journal of Advanced Robotic Systems*. https://doi.org/10.5772/5618

Mohan, S., Klenk, M., Shreve, M., Evans, K., Ang, A., & Maxwell, J. (2020, July 29). Characterizing an Analogical Concept Memory for Architectures Implementing the Common Model of Cognition. *Advances in Cognitive Systems*. http://arxiv.org/abs/2006.01962

Mohan, S., & Laird, J. (2014). Learning goal-oriented hierarchical tasks from situated interactive instruction. *Proceedings of the AAAI Conference on Artificial Intelligence*, *28*(1). https://ojs.aaai.org/index.php/AAAI/article/view/8756

Molineaux, M., Klenk, M., & Aha, D. (2010). Goal-Driven Autonomy in a Navy Strategy Simulation. *Proceedings of the AAAI Conference on Artificial Intelligence*, *24*(1), 1548–1554. https://doi.org/10.1609/aaai.v24i1.7576

Moratz, R., & Tenbrink, T. (2006). Spatial Reference in Linguistic Human-Robot Interaction: Iterative, Empirically Supported Development of a Model of Projective Relations. *Spatial Cognition & Computation*, *6*(1), 63–107. https://doi.org/10.1207/s15427633scc0601_3

Moser, E. I., Roudi, Y., Witter, M. P., Kentros, C., Bonhoeffer, T., & Moser, M.-B. (2014). Grid cells and cortical representation. *Nature Reviews Neuroscience*, *15*(7), 466–481. https://doi.org/10.1038/nrn3766

Nakos, C., Rabkina, I., Hill, S., & Forbus, K. D. (2020). *Corrective Processes in Modeling Reference Resolution: 42nd Annual Meeting of the Cognitive Science Society: Developing a Mind: Learning in Humans, Animals, and Machines, CogSci 2020*. 171–177. http://www.scopus.com/inward/record.url?scp=85139480102&partnerID=8YFLogxK

Nuxoll, A., & Laird, J. (2007, January 1). Extending Cognitive Architecture with Episodic Memory. *AAAI*.

Nuxoll, A., & Laird, J. E. (2004). *A Cognitive Model of Episodic Memory Integrated With a General Cognitive Architecture*. 6.

Nuxoll, A. M., & Laird, J. E. (2012). Enhancing intelligent agents with episodic memory. *Cognitive Systems Research*, *17–18*, 34–48. https://doi.org/10.1016/j.cogsys.2011.10.002

Ontañón, S., Mishra, K., Sugandh, N., & Ram, A. (2007). Case-Based Planning and Execution for Real-Time Strategy Games. In R. O. Weber & M. M. Richter (Eds.), *Case-Based Reasoning Research and Development* (Vol. 4626, pp. 164–178). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-74141-1_12

Ontañón, S., & Ram, A. (2011). Case-Based Reasoning and User-Generated Artificial Intelligence for Real-Time Strategy Games. In P. A. González-Calero & M. A. Gómez-Martín (Eds.), *Artificial Intelligence for Computer Games* (pp. 103–124). Springer. https://doi.org/10.1007/978-1-4419-8188-2_5

OpenAI, Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J.,

Petrov, M., Pinto, H. P. d O., Raiman, J., Salimans, T., … Zhang, S. (2019). *Dota 2 with Large Scale Deep Reinforcement Learning* (arXiv:1912.06680). arXiv. http://arxiv.org/abs/1912.06680

Parisi, G. I., Tani, J., Weber, C., & Wermter, S. (2018). Lifelong Learning of Spatiotemporal Representations with Dual-Memory Recurrent Self-Organization. *Frontiers in Neurorobotics*, *12*, 78. https://doi.org/10.3389/fnbot.2018.00078

Quinlan, J. R. (1990). Learning logical definitions from relations. *Machine Learning*, *5*(3), 239–266. https://doi.org/10.1007/BF00117105

Rabkina, I., McFate, C., & Forbus, K. D. (2018). *Bootstrapping from Language in the Analogical Theory of Mind Model*. 6.

Radvansky, G. A., Krawietz, S. A., & Tamplin, A. K. (2011). Walking through doorways causes forgetting: Further explorations. *Quarterly Journal of Experimental Psychology (2006)*, *64*(8), 1632–1645. https://doi.org/10.1080/17470218.2011.571267

Radvansky, G. A., & Zacks, J. M. (2017). Event Boundaries in Memory and Cognition. *Current Opinion in Behavioral Sciences*, *17*, 133–140. https://doi.org/10.1016/j.cobeha.2017.08.006

Radvansky, G., O'Rear, A., & Fisher, J. (2017). Event models and the fan effect. *Memory & Cognition*, *45*. https://doi.org/10.3758/s13421-017-0713-4

Schank, R. C. (1980). Language and Memory. *Cognitive Science*, *4*(3), 243–284. https://doi.org/10.1207/s15516709cog0403_2

She, L., Yang, S., Cheng, Y., Jia, Y., Chai, J., & Xi, N. (2014). Back to the Blocks World: Learning New Actions through Situated Human-Robot Dialogue. In K. Georgila, M. Stone, H. Hastie, & A. Nenkova (Eds.), *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)* (pp. 89–97). Association for Computational Linguistics. https://doi.org/10.3115/v1/W14-4313

Song, C. H., Wu, J., Washington, C., Sadler, B. M., Chao, W.-L., & Su, Y. (2023). *LLM-Planner: Few-Shot Grounded Planning for Embodied Agents with Large Language Models* (arXiv:2212.04088). arXiv. https://doi.org/10.48550/arXiv.2212.04088

Sridhar, M., Cohn, A., & Hogg, D. (2008). Learning Functional Object-Categories from a Relational Spatio-Temporal Representation. In *ECAI* (p. 610). https://doi.org/10.3233/978-1-58603-891-5-606

Sridhar, M., Cohn, A., & Hogg, D. (2010). Unsupervised Learning of Event Classes from Video. *Proceedings of the AAAI Conference on Artificial Intelligence*, *24*, Article 1. https://doi.org/10.1609/aaai.v24i1.7726

Swallow, K., Zacks, J., & Abrams, R. (2009). Event Boundaries in Perception Affect Memory Encoding and Updating. *Journal of Experimental Psychology. General*, *138*, 236–257. https://doi.org/10.1037/a0015631

Tenorth, M., & Beetz, M. (2013). KnowRob: A knowledge processing infrastructure for cognition-enabled robots. *The International Journal of Robotics Research*, *32*(5), 566–590. https://doi.org/10.1177/0278364913481635

Tulving, E. (1983). *Elements of Episodic Memory*. Clarendon Press.

Tulving, E. (2002). Episodic memory: From mind to brain. *Annual Review of Psychology*, *53*(1), 1–25.

Ulam, P., Jones, J., & Goel, A. (2008). Combining Model-Based Meta-Reasoning and Reinforcement Learning For Adapting Game-Playing Agents. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, *4*(1), 132–137. https://doi.org/10.1609/aiide.v4i1.18685

van de Weghe, N., Cohn, A., Tré, G., & De Maeyer, P. (2006). A qualitative trajectory calculus as a basis for representing moving objects in Geographical Information Systems. *CONTROL AND CYBERNETICS*, *35*, 97–119.

Wang, J., Wu, Z., Li, Y., Jiang, H., Shu, P., Shi, E., Hu, H., Ma, C., Liu, Y., Wang, X., Yao, Y., Liu, X., Zhao, H., Liu, Z., Dai, H., Zhao, L., Ge, B., Li, X., Liu, T., & Zhang, S. (2024). *Large Language Models for Robotics: Opportunities, Challenges, and Perspectives* (arXiv:2401.04334). arXiv. https://doi.org/10.48550/arXiv.2401.04334

Weber, B. G., & Mateas, M. (2009). Conceptual Neighborhoods for Retrieval in Case-Based Reasoning. In L. McGinty & D. C. Wilson (Eds.), *Case-Based Reasoning Research and Development* (Vol. 5650, pp. 343–357). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-02998-1_25

Weber, B. G., Mateas, M., & Jhala, A. (2010). *Applying Goal-Driven Autonomy to StarCraft*. 6.

Weber, B., & Mateas, M. (2009). Case-Based Reasoning for Build Order in Real-Time Strategy Games. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, *5*(1), Article 1. https://doi.org/10.1609/aiide.v5i1.12360

Wetzel, B., Anderson, K., Koutstaal, W., & Gini, M. (2012). If Not Now, Where? Time and Space Equivalency in Strategy Games. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, *8*(1), Article 1. https://doi.org/10.1609/aiide.v8i1.12506

Wiley, T., Sammut, C., & Bratko, I. (2013). *Planning with Qualitative Models for Robotic Domains*.

Wiley, T., Sammut, C., Hengst, B., & Bratko, I. (2016). *A Planning and Learning Hierarchy using Qualitative Reasoning for the Online Acquisition of Robotic Behaviors*.

Wolter, D., & Kirsch, A. (2015). *Qualitative Spatial Reasoning for Boosting Learning-Based Robotics*.

Wood, J. N., & Spelke, E. S. (2005). Infants' enumeration of actions: Numerical discrimination and its signature limits. *Developmental Science*, *8*(2), 173–181. https://doi.org/10.1111/j.1467-7687.2005.00404.x

Worboys, M., & Duckham, M. (2006). Monitoring qualitative spatiotemporal change for geosensor networks. *International Journal of Geographical Information Science*, *20*(10), 1087–1108. https://doi.org/10.1080/13658810600852180

Wu, X., Chakraborty, S., Xian, R., Liang, J., Guan, T., Liu, F., Sadler, B. M., Manocha, D., & Bedi, A. S. (2025). *On the Vulnerability of LLM/VLM-Controlled Robotics* (arXiv:2402.10340). arXiv. https://doi.org/10.48550/arXiv.2402.10340

Zacks, J. M. (2020). Event Perception and Memory. *Annual Review of Psychology*, *71*, 165–191. https://doi.org/10.1146/annurev-psych-010419-051101

Zacks, J. M., & Tversky, B. (2001). Event Structure in Perception and Conception. *Psychological Bulletin*, *127*(1), 3–21. https://doi.org/10.1037/0033-2909.127.1.3

Zhou, J., Ye, K., Liu, J., Ma, T., Wang, Z., Qiu, R., Lin, K.-Y., Zhao, Z., & Liang, J. (2025). *Exploring the Limits of Vision-Language-Action Manipulations in Cross-task Generalization* (arXiv:2505.15660). arXiv. https://doi.org/10.48550/arXiv.2505.15660

APPENDIX A: ALGORITHMS AND KNOWLEDGE

Algorithm 2: Determine persistence between compound spatial entities.

Input: **CSE-before**: a set of compound spatial entities at time t_{n-1}

CSE-after: a set of compound spatial entities at time t_n

Output: **Persist-Tuples**: a list of 2-tuples, [(before-name, after-name) …] indicating that the entity before-name in CSE-before persists as after-name in CSE-after

```
 1: DETERMINE-PERSISTENCE (CSE-before, CSE-after)
 2: Let g ← GENERATE-GRAPH(CSE-before, CSE-after)
 3: Let Persist-Tuples ← empty list
 4: for each CC ∈ connected-components(g)
 5:       Let max-prev ← max_size(U(CC))
 6:       Let max-after ← max-size(V(CC))
 7:       add list(max-prev, max-after) to Persist-Tuples
 8: end for each
 9: return Persist-Tuples
10:
11: GENERATE-GRAPH(CSE-before, CSE-after)
12: /* Generates a directed bipartite graph from nodes in CSE-before (U) to nodes in CSE-after
    (V). Arcs indicate temporal correspondence between compound entities (nodes). */
13: Let g be an empty directed graph
14: for each PE ∈ CSE-before
15:  for each SE ∈ CSE-after
16:       if CORRESPONDENCE(PE, CE)
17:            add-arc(g, PE, CE)
18:       endif
19:  end for each
20: end for each
21: return g
22:
23: CORRESPONDENCE(PE, CE)
24: /* determine correspondence between previous and current compound entities, PE, CE */
25: return set-intersection(participants(PE), participants(CE))
```

Algorithm 3: QSTEM Case Construction

Input: **EVENT-INTS**: Top-level event intervals

OBJECT-INTS: Latest-first lists of top-level object intervals indexed by entity name

ENTS: a set of participant entities

QINTS: a set of qualitative intervals describing change for entities

Output: QSTEM-CASE

```
CONSTRUCT-CASE (event-ints, object-ints, ents, qints)
Let case ← empty set of facts
add-successor-relations(event-ints, case)
for each event-int ∈ event-ints
    reify-interval(event-int, case)
end for each
for each entity ∈ ents
    reify-entity-history(entity, qtypes, qints, event-ints, object-ints, case)
end for each
return case

REIFY-ENTITY-HISTORY(entity, qtypes, qints, event-ints, object-ints, case)
Let object-ints ← object-ints[entity]
Let entity-intervals ← qints[entity]
add-entity-facts(entity, case)
add-successor-relations(object-ints, case)
add-successor-relations(entity-intervals, case)
for each oint ∈ object-ints
    reify-interval(oint, case)
    for each eint ∈ event-ints
        if connected(oint, eint)
            add-fact(allen-relation(eint, oint), case)
        end if
    end for each
    for each qint ∈ entity-intervals
        if connected(oint, qint)
            add-fact(allen-relation(oint, qint), case)
        end if
    end for each
end for each
return case

ADD-SUCCESSOR-RELATIONS(intervals, case)
for each (before, after) pair in intervals
    add-fact('(meets ,before ,after), case)
end for each

CONNECTED(int-a, int-b)
return not member(allen-relation(int-a, int-b), [precedes, precededBy])

REIFY-INTERVAL(int, case)
```

```
43:  /* e.g. (nonDecreasingIn Int-1 (Defenders NYC-Footprint)) /*
43:  Let fact ← (list (quantity-predicate int) (name int) (quantity-type int))
44:  add-fact(fact, case)
```

Algorithm 4: Decision making for a city from prior battles

Input: **ENTITY**: The city making a decision

QSTEM-CASE: Constructed case for city

GOAL-NETWORK: The qualitative goal network

QUALITATIVE-MODEL: The Freeciv qualitative model

Output: production decision

```
 1:  MAKE-DECISION (entity, qstem-case, goal-network, qualitative-model)
 2:  if not GROWTH-PHASE
 3:      Let decisions ← empty list
 4:      Let probe ← CONSTRUCT-CASE(entity)
 5:      Let target, mapping ← SAGE_CLASSIFY(probe, [SUCCESS-GPOOL, FAILURE-
         GPOOL])
 6:      if target is FAILURE-GPOOL
 7:          Let subgoals ← subgoals(goal-network[CITY-MILSYS-NODE])
 8:          for each goal ∈ subgoals
 9:              if determine-failure(entity, goal)
10:                  Let action ← propose-action-influence(goal, qualitative-model)
11:                  add action to decisions
12:              end if
13:          end for each
14:          return random-selection(decisions)
15:      end if
16:  end if
17:  return default-decision-procedure(entity)
18:
19:  DETERMINE-FAILURE(entity, goal, mapping)
20:  if achieve-goal-p(goal)
21:      return satisfies(city, goal)
22:  else if quantity-constraint-goal-p(goal)
23:      Let quantity-type ← goal-to-quantity-type(goal)
24:      Let cur-val ← current-value(quantity-type, entity)
25:      Let prev-vals ← previous-values(entity, quantity-type, cur-val, mapping)
26:      if min-goal-p(goal)
27:          return cur-val ≥ min(prev-vals)
28:      else if max-goal-p(goal)
29:          return cur-val ≤ max(prev-vals)
```

30: **end if**
31: **end if**
32:
33: **GOAL-TO-QUANTITY-TYPE(goal)**
34: /* The goal form for quantity-constraint goals is (MaximizeFn <?quantity-type>) or (MinimizeFn <?quantity-type>), so this returns <?quantity-type> */
35:
36: **PROPOSE-ACTION-INFLUENCE(goal, qualitative-model)**
37: /* The qualitative model describes influences for both achieve and quantity constraint goals. For achieve goals, an example is 'producing city walls results in the city having walls. For quantity constraint goals, an example is 'producing archers increases the number of military units. Takes a goal (node from the goal network) and a qualitative model as input. Outputs a set of actions that positively influence that goal. */
38:
39: **PREVIOUS-VALUES(entity, quantity-type, mapping)**
40: /* Here, mapping contains knowledge about correspondences between facts in the probe case and facts in the target. For example, the qpred
(Defenders (AtFn (HistoryFn NYC-Footprint) (StartFn CUREVENT)) ?val)
might map to the target-qpred
(Defenders (AtFn (HistoryFn (GEFn 1)) (StartFn (GEFn 2))) (GEFn 3)).
Here, (GEFn 3) has an associated list of quantity values from cases that participate in the generalization, e.g. ((UnitCount 1) (UnitCount 0) (UnitCount 3)), which are returned by the prev-values function */
41: Let qpred ← ((,quantity-type (AtFn (HistoryFn ?entity) (StartFn CUREVENT))) ?val)
42: Let target-qpred ← correspondence-target(qpred, mapping)
43: return prev-values(target-qpred)

9.1 Freeciv Definitions

Table 5: Quantity definitions and associated encodings for Freeciv experiments

Quantity	Description	Encodings
CivSize	Area of the player's national footprint	DS
DefendingUnitCount	Number of friendly military units within a city footprint	MS, DS
PlayerGold	Amount of gold for a player	DS
CitySize	Size of a city (population estimate given by Freeciv engine)	DS
UnitGroupSize	Number of participants in unit group footprint	DS
UnitGroupArea	Area of unit group foootprint	DS
Attackers	Aggregate number of enemy units on battlefield	MS, DS

Table 6: Attribute listing for Freeciv entities

Entity	Description
Player	(isa <?player> Freeciv-Player)
	(currentGovernment <?player> <?current-gov>)
	(footprintOf <?player> <?player's national footprint>)
City	(isa <?city> Freeciv-City)
	(cityOwner <?city> <?player>)
	(cityHasImprovement <?city> <?improvement>)
	(footprintOf <?city> <?city-footprint>)
Unit	(isa <?unit> Freeciv-Unit)
	(unitOwner <?unit> <?player>)
All Footprints	(isa <?footprint> <?footprint-type>)
National Footprints	(isa <?footprint> FC-Region-EnemyTerritory) # when controlled by opponent
	(isa <?footprint> FC-Region-FriendlyTerritory # when controlled by ally
UnitGroupFootprint	(isa <?footprint> EnemyFootprint) # when controlled by opponent
ContinentBlob	;;; from (McLure and Forbus, 2012)
	(isa <?blob> ContinentBlob)

APPENDIX B: EXAMPLE CASE FROM FREECIV EXPERIMENT ONE

(in-microtheory (DecisionCaseFn (CommandFn fc-game1 163) (doChangeProduction FC-City-Salto FC-Unit-Settlers)))

;;; glossed, now facts like (meets ?interval1 ?interval2) contain descriptions of the intervals (e.g. ;;; increasing gold for player) instead of reified entities

(cityGrowthRate FC-City-Salto (Percent 5))

(cityOwner FC-City-Butare FC-Player-MutaraIII)

(cityOwner FC-City-Salto FC-Player-JosÃ©Artigas)

(citySize FC-City-Butare (FreecivCitizenCountFn 1))

(citySize FC-City-Butare (FreecivCitizenCountFn 2))

(citySize FC-City-Butare (FreecivCitizenCountFn 3))

(citySize FC-City-Salto (FreecivCitizenCountFn 1))

(currentGold FC-Player-JosÃ©Artigas (GoldPoints 99))

(currentGold FC-Player-MutaraIII (GoldPoints 0))

(currentGovernment FC-Player-JosÃ©Artigas FC-Despotism)

(currentGovernment FC-Player-MutaraIII FC-Anarchy)

(footprintOf FC-City-Butare CityFootprint-FC-City-Butare)

(footprintOf FC-City-Salto CityFootprint-FC-City-Salto)

(footprintOf FC-Player-JosÃ©Artigas TerritoryBlob-0)

(footprintOf FC-Player-MutaraIII TerritoryBlob-1)

(isa CityFootprint-FC-City-Butare CityFootprint)

(isa CityFootprint-FC-City-Salto CityFootprint)

(isa ContinentBlob-1 FreeCiv-TerrainBlob-Water)

(isa ContinentBlob-2 FreeCiv-TerrainBlob-Land)

(isa ContinentBlob-4 FreeCiv-TerrainBlob-Water)

(isa FC-City-Butare FreeCiv-City)

(isa FC-City-Salto FreeCiv-City)

(isa FC-Player-JosÃ©Artigas FreeCiv-Player)

(isa FC-Player-MutaraIII FreeCiv-Player)

(isa TerritoryBlob-0 FC-Region-OurOwnTerritory)

(isa TerritoryBlob-0 NationalTerritoryFootprint)

(isa TerritoryBlob-1 FC-Region-EnemyTerritory)

(isa TerritoryBlob-1 NationalTerritoryFootprint)

(isa TerritoryBlob-92 NationalTerritoryFootprint)

(isa TerritoryBlob-93 NationalTerritoryFootprint)

(numTurnsToBuild FC-City-Salto (Turns 20))

(numTurnsToGrow FC-City-Salto (Turns 10))

(rcc8-EC CityFootprint-FC-City-Salto CityFootprint-FC-City-Butare)

(rcc8-EC CityFootprint-FC-City-Salto TerritoryBlob-1)

(rcc8-EC CityFootprint-FC-City-Salto TerritoryBlob-92)

(rcc8-EC CityFootprint-FC-City-Salto TerritoryBlob-93)

(rcc8-NTPP CityFootprint-FC-City-Salto ContinentBlob-1)

(rcc8-PO CityFootprint-FC-City-Salto ContinentBlob-2)

(rcc8-PO CityFootprint-FC-City-Salto ContinentBlob-4)

(rcc8-TPP CityFootprint-FC-City-Salto TerritoryBlob-0)

(starts-after (decreasing citySize FC-City-Salto) (Zero numPresentMilitaryUnits FC-City-Salto))

(starts-at-same-time (increasing nationalFootprintSize FC-Player-JosÃ©Artigas)

  (increasing currentGold FC-Player-JosÃ©Artigas))

(unitOwner Unit-148 FC-Player-JosÃ©Artigas)

APPENDIX C: EXAMPLE CASE FROM FREECIV EXPERIMENT TWO

((Area (AtFn (EndFn STEntity-372) STEntity-372)) (FreecivTileArea 14))

((Area (AtFn (EndFn STEntity-372) STEntity-372)) (FreecivTileArea 8))

((Area (AtFn (StartFn STEntity-372) STEntity-372)) (FreecivTileArea 14))

((Area (AtFn (StartFn STEntity-372) STEntity-372)) (FreecivTileArea 8))

((BattleOpposingUnitCardinality (AtFn (EndFn STEntity-372) STEntity-372))
(FreecivUnitCountFn 2))

((BattleOpposingUnitCardinality (AtFn (StartFn STEntity-372) STEntity-372))
(FreecivUnitCountFn 2))

((Defenders (AtFn (EndFn STEntity-372) STEntity-372)) (FreecivUnitCountFn 0))

((Defenders (AtFn (EndFn STEntity-372) STEntity-372)) (FreecivUnitCountFn 1))

((Defenders (AtFn (EndFn STEntity-372) STEntity-372)) (FreecivUnitCountFn 2))

((Defenders (AtFn (StartFn STEntity-372) STEntity-372)) (FreecivUnitCountFn 0))

((Defenders (AtFn (StartFn STEntity-372) STEntity-372)) (FreecivUnitCountFn 1))

((Defenders (AtFn (StartFn STEntity-372) STEntity-372)) (FreecivUnitCountFn 2))

((citySize (AtFn (EndFn STEntity-372) STEntity-372)) (FreecivCitizenCountFn 1))

((citySize (AtFn (StartFn STEntity-372) STEntity-372)) (FreecivCitizenCountFn 1))

(cityOwner FC-City-Chengdu FC-Player-DengXiaoping)

(contains (constant footprintParticipantCardinality UnitGroupFootprint-2070897) (increasing blobArea UnitGroupFootprint-2070897))

(currentActivity Unit-292 FC-ActivityIdle)

(fcObjectAt FC-City-Chengdu (FreecivLocationFn 36 27))

(fcObjectAt Unit-292 (FreecivLocationFn 36 27))

(footprintOf FC-City-Chengdu CityFootprint-FC-City-Chengdu)

(isa CityFootprint-FC-City-Chengdu CityFootprint)

(isa FC-City-Chengdu FreeCiv-City)

(isa STEntity-372 DefensiveEpisodeFailure)

(isa STEntity-372 SpatiotemporalEvent)

(isa Unit-292 FC-Unit-Catapult)

(isa UnitGroupFootprint-1409544 UnitGroupFootprint)

(isa UnitGroupFootprint-1638551 UnitGroupFootprint)

(isa UnitGroupFootprint-2070897 UnitGroupFootprint)

(meets-inverse (constant footprintParticipantCardinality UnitGroupFootprint-1638551) (increasing blobArea UnitGroupFootprint-1638551))

(meets-inverse (decreasing footprintParticipantCardinality UnitGroupFootprint-1409544) (increasing blobArea UnitGroupFootprint-1409544))

(meets-inverse (constant blobArea UnitGroupFootprint-1638551) (increasing blobArea UnitGroupFootprint-1638551))

(meets-inverse (decreasing blobArea UnitGroupFootprint-1409544) (increasing blobArea UnitGroupFootprint-1409544))

(meets-inverse (decreasing blobArea UnitGroupFootprint-2070897) (increasing blobArea UnitGroupFootprint-2070897))

(meets-inverse (increasing blobArea UnitGroupFootprint-1409544) (constant blobArea UnitGroupFootprint-1409544))

(meets-inverse (increasing blobArea UnitGroupFootprint-1638551) (constant blobArea UnitGroupFootprint-1638551))

(meets-inverse (increasing blobArea UnitGroupFootprint-2070897) (constant blobArea UnitGroupFootprint-2070897))

(overlapped-by (constant footprintParticipantCardinality UnitGroupFootprint-1638551) (constant blobArea UnitGroupFootprint-1638551))

(overlapped-by (decreasing footprintParticipantCardinality UnitGroupFootprint-1409544) (constant blobArea UnitGroupFootprint-1409544))

(started-by (constant footprintParticipantCardinality UnitGroupFootprint-2070897) (constant blobArea UnitGroupFootprint-2070897))

(starts-at-same-time (constant footprintParticipantCardinality UnitGroupFootprint-1638551) (constant blobArea UnitGroupFootprint-1638551))

(starts-at-same-time (decreasing footprintParticipantCardinality UnitGroupFootprint-1409544) (decreasing blobArea UnitGroupFootprint-1409544))

(starts-at-same-time (constant citySize FC-City-Chengdu) (constant numMilitaryUnitsInBlob CityFootprint-FC-City-Chengdu))

(starts-at-same-time (constant citySize FC-City-Chengdu) (zero numMilitaryUnitsInBlob CityFootprint-FC-City-Chengdu))

(starts-before (constant footprintParticipantCardinality UnitGroupFootprint-2070897) (decreasing blobArea UnitGroupFootprint-2070897))

(unitOwner Unit-292 FC-Player-GiacomoMatteotti)

APPENDIX D: EXAMPLE SPREAD CASE

;;; glossed

(in-microtheory (AileenEventMtFn HistoryCondition spread spread-1d7f970c) :exclude-globals t)

(during (changing PositionFn STRegion-3499) (changing PositionFn hand))

(during (changing PositionFn STRegion-81e4) (constant PositionFn hand))

(during (changing PositionFn STRegion-8adb) (changing PositionFn hand))

(during (changing PositionFn STRegion-d7e1) (constant HeightFn STRegion-d7e1))

(during (changing PositionFn STRegion-d7e1) (constant LeftRightPositionFn STRegion-d7e1))

(during (changing PositionFn STRegion-d7e1) (holdsIn tpp hand STRegion-d7e1))

(during (changing PositionFn STRegion-d7e1) (increasing ForwardBackwardPositionFn STRegion-d7e1))

(during (changing PositionFn STRegion-d7e1) (increasing VolumeFn STRegion-d7e1))

(during (changing PositionFn STRegion-d7e1) (constant PositionFn hand))

(during (changing PositionFn hand) PARENT_EPISODE)

(during (constant PositionFn STRegion-790e) (constant PositionFn hand))

(during (constant PositionFn STRegion-81e4) (constant PositionFn hand))

(during (constant PositionFn STRegion-8adb) (changing PositionFn hand))

(during (constant PositionFn STRegion-8bbc) (constant PositionFn hand))

(during (constant PositionFn STRegion-938a) (constant VolumeFn STRegion-938a))

(during (constant PositionFn STRegion-938a) (holdsIn dc hand STRegion-938a))

(during (constant PositionFn STRegion-938a) (constant PositionFn hand))

(during (constant PositionFn STRegion-ab5f) (constant PositionFn hand))

(during (constant PositionFn STRegion-d7e1) (constant HeightFn STRegion-d7e1))

(during (constant PositionFn STRegion-d7e1) (constant LeftRightPositionFn STRegion-d7e1))

(during (constant PositionFn STRegion-d7e1) (holdsIn tpp hand STRegion-d7e1))

(during (constant PositionFn STRegion-d7e1) (increasing DistanceFn hand))

(during (constant PositionFn STRegion-d7e1) (increasing ForwardBackwardPositionFn STRegion-d7e1))

(during (constant PositionFn STRegion-d7e1) (increasing VolumeFn STRegion-d7e1))

(during (constant PositionFn STRegion-d7e1) (constant PositionFn hand))

(during (constant PositionFn hand) PARENT_EPISODE)

(finished-by (changing PositionFn STRegion-1adb) (increasing LeftRightPositionFn STRegion-1adb))

(finished-by (constant PositionFn STRegion-938a) (constant DistanceFn hand))

(finished-by (constant PositionFn STRegion-938a) (constant ForwardBackwardPositionFn STRegion-938a))

(finished-by (constant PositionFn STRegion-938a) (constant LeftRightPositionFn STRegion-938a))

(finished-by (constant PositionFn STRegion-d7e1) (constant PositionFn hand))

(finishes (changing PositionFn STRegion-938a) (constant PositionFn hand))

(finishes (changing PositionFn STRegion-938a) (constant VolumeFn STRegion-938a))

(finishes (changing PositionFn STRegion-938a) (holdsIn dc hand STRegion-938a))

(finishes (changing PositionFn STRegion-ab5f) (constant DistanceFn hand))

(finishes (changing PositionFn STRegion-ab5f) (constant ForwardBackwardPositionFn STRegion-ab5f))

(finishes (changing PositionFn STRegion-ab5f) (constant HeightFn STRegion-ab5f))

(finishes (changing PositionFn STRegion-ab5f) (constant PositionFn hand))

(finishes (changing PositionFn STRegion-ab5f) (constant VolumeFn STRegion-ab5f))

(finishes (changing PositionFn STRegion-ab5f) (holdsIn ec hand STRegion-ab5f))

(finishes (changing PositionFn STRegion-d7e1) (increasing DistanceFn hand))

(finishes (changing PositionFn STRegion-d7e1) (increasing ForwardBackwardPositionFn STRegion-d7e1))

(finishes (changing PositionFn STRegion-d7e1) (increasing VolumeFn STRegion-d7e1))

(finishes (constant PositionFn STRegion-1adb) (constant HeightFn STRegion-1adb))

(finishes (constant PositionFn STRegion-1adb) (constant PositionFn hand))

(finishes (constant PositionFn STRegion-1adb) (constant VolumeFn STRegion-1adb))

(finishes (constant PositionFn STRegion-1adb) (holdsIn dc hand STRegion-1adb))

(finishes (constant PositionFn STRegion-790e) (constant HeightFn STRegion-790e))

(finishes (constant PositionFn STRegion-790e) (constant VolumeFn STRegion-790e))

(finishes (constant PositionFn STRegion-790e) (holdsIn dc hand STRegion-790e))

(finishes (constant PositionFn STRegion-790e) (increasing DistanceFn hand))

(finishes (constant PositionFn STRegion-81e4) (constant ForwardBackwardPositionFn STRegion-81e4))

(finishes (constant PositionFn STRegion-81e4) (constant HeightFn STRegion-81e4))

(finishes (constant PositionFn STRegion-81e4) (constant LeftRightPositionFn STRegion-81e4))

(finishes (constant PositionFn STRegion-81e4) (constant VolumeFn STRegion-81e4))

(finishes (constant PositionFn STRegion-81e4) (holdsIn po hand STRegion-81e4))

(finishes (constant PositionFn STRegion-81e4) (increasing DistanceFn hand))

(finishes (constant PositionFn STRegion-8adb) (constant HeightFn STRegion-8adb))

(finishes (constant PositionFn STRegion-8adb) (constant LeftRightPositionFn STRegion-8adb))

(finishes (constant PositionFn STRegion-8adb) (constant VolumeFn STRegion-8adb))

(finishes (constant PositionFn STRegion-8adb) (holdsIn po hand STRegion-8adb))

(finishes (constant PositionFn STRegion-8adb) (increasing DistanceFn hand))

(finishes (constant PositionFn STRegion-938a) (constant HeightFn STRegion-938a))

(finishes (constant PositionFn STRegion-d7e1) (constant HeightFn STRegion-d7e1))

(finishes (constant PositionFn STRegion-d7e1) (constant LeftRightPositionFn STRegion-d7e1))

(finishes (constant PositionFn STRegion-d7e1) (constant PositionFn hand))

(finishes (constant PositionFn STRegion-d7e1) (holdsIn tpp hand STRegion-d7e1))

(finishes (constant PositionFn hand) PARENT_EPISODE)

(isa AILEEN CognitiveRobot)

(isa INT4224 PARENT_EPISODE)

(isa hand Hand)

(meets (changing PositionFn STRegion-1adb) (constant ForwardBackwardPositionFn STRegion-1adb))

(meets (changing PositionFn STRegion-1adb) (constant LeftRightPositionFn STRegion-1adb))

(meets (changing PositionFn STRegion-1adb) (constant PositionFn STRegion-1adb))

(meets (changing PositionFn STRegion-1adb) (constant DistanceFn hand))

(meets (changing PositionFn STRegion-790e) (constant ForwardBackwardPositionFn STRegion-790e))

(meets (changing PositionFn STRegion-790e) (constant LeftRightPositionFn STRegion-790e))

(meets (changing PositionFn STRegion-790e) (constant PositionFn STRegion-790e))

(meets (changing PositionFn STRegion-81e4) (constant PositionFn STRegion-81e4))

(meets (changing PositionFn STRegion-8adb) (constant ForwardBackwardPositionFn STRegion-8adb))

(meets (changing PositionFn STRegion-8adb) (constant PositionFn STRegion-8adb))

(meets (changing PositionFn STRegion-938a) (constant PositionFn STRegion-938a))

(meets (changing PositionFn STRegion-d7e1) (constant ForwardBackwardPositionFn STRegion-d7e1))

(meets (changing PositionFn STRegion-d7e1) (constant PositionFn STRegion-d7e1))

(meets (changing PositionFn STRegion-d7e1) (constant VolumeFn STRegion-d7e1))

(meets (changing PositionFn STRegion-d7e1) (constant DistanceFn hand))

(meets (changing PositionFn hand) (constant PositionFn hand))

(meets (constant DistanceFn hand) (decreasing DistanceFn hand))

(meets (constant DistanceFn hand) (increasing DistanceFn hand))

(meets (constant ForwardBackwardPositionFn STRegion-938a) (increasing ForwardBackwardPositionFn STRegion-938a))

(meets (constant ForwardBackwardPositionFn STRegion-d7e1) (increasing ForwardBackwardPositionFn STRegion-d7e1))

(meets (constant HeightFn STRegion-938a) (decreasing HeightFn STRegion-938a))

(meets (constant LeftRightPositionFn STRegion-938a) (increasing LeftRightPositionFn STRegion-938a))

(meets (constant LeftRightPositionFn STRegion-ab5f) (increasing LeftRightPositionFn STRegion-ab5f))

(meets (constant PositionFn STRegion-938a) (changing PositionFn STRegion-938a))

(meets (constant PositionFn STRegion-938a) (decreasing HeightFn STRegion-938a))

(meets (constant PositionFn STRegion-938a) (increasing ForwardBackwardPositionFn STRegion-938a))

(meets (constant PositionFn STRegion-938a) (increasing LeftRightPositionFn STRegion-938a))

(meets (constant PositionFn STRegion-938a) (increasing DistanceFn hand))

(meets (constant PositionFn STRegion-ab5f) (changing PositionFn STRegion-ab5f))

(meets (constant PositionFn STRegion-ab5f) (increasing LeftRightPositionFn STRegion-ab5f))

(meets (constant PositionFn STRegion-d7e1) (changing PositionFn STRegion-d7e1))

(meets (constant PositionFn STRegion-d7e1) (changing PositionFn hand))

(meets (constant PositionFn hand) (changing PositionFn hand))

(meets (constant VolumeFn STRegion-d7e1) (increasing VolumeFn STRegion-d7e1))

(meets (decreasing DistanceFn hand) (increasing DistanceFn hand))

(meets (decreasing LeftRightPositionFn STRegion-1adb) (increasing LeftRightPositionFn STRegion-1adb))

(meets (holdsIn ec hand STRegion-1adb) (holdsIn dc hand STRegion-1adb))

(meets (holdsIn ec hand STRegion-790e) (holdsIn dc hand STRegion-790e))

(meets (holdsIn ec hand STRegion-938a) (holdsIn dc hand STRegion-938a))

(meets (holdsIn ec hand STRegion-d7e1) (holdsIn po hand STRegion-d7e1))

(meets (holdsIn po hand STRegion-d7e1) (holdsIn tpp hand STRegion-d7e1))

(meets (increasing DistanceFn hand) (constant DistanceFn hand))

(meets (increasing ForwardBackwardPositionFn STRegion-1adb) (constant ForwardBackwardPositionFn STRegion-1adb))

(meets (increasing ForwardBackwardPositionFn STRegion-790e) (constant ForwardBackwardPositionFn STRegion-790e))

(meets (increasing ForwardBackwardPositionFn STRegion-8adb) (constant ForwardBackwardPositionFn STRegion-8adb))

(meets (increasing ForwardBackwardPositionFn STRegion-938a) (constant ForwardBackwardPositionFn STRegion-938a))

(meets (increasing ForwardBackwardPositionFn STRegion-d7e1) (constant ForwardBackwardPositionFn STRegion-d7e1))

(meets (increasing LeftRightPositionFn STRegion-1adb) (constant LeftRightPositionFn STRegion-1adb))

(meets (increasing LeftRightPositionFn STRegion-790e) (constant LeftRightPositionFn STRegion-790e))

(meets (increasing LeftRightPositionFn STRegion-938a) (constant LeftRightPositionFn STRegion-938a))

(meets (increasing VolumeFn STRegion-d7e1) (constant VolumeFn STRegion-d7e1))

(meets-inverse (changing PositionFn STRegion-938a) (constant DistanceFn hand))

(meets-inverse (changing PositionFn STRegion-938a) (constant ForwardBackwardPositionFn STRegion-938a))

(meets-inverse (changing PositionFn STRegion-938a) (constant HeightFn STRegion-938a))

(meets-inverse (changing PositionFn STRegion-938a) (constant LeftRightPositionFn STRegion-938a))

(meets-inverse (changing PositionFn STRegion-ab5f) (constant LeftRightPositionFn STRegion-ab5f))

(meets-inverse (changing PositionFn STRegion-d7e1) (constant PositionFn hand))

(meets-inverse (constant PositionFn STRegion-1adb) (increasing DistanceFn hand))

(meets-inverse (constant PositionFn STRegion-1adb) (increasing ForwardBackwardPositionFn STRegion-1adb))

(meets-inverse (constant PositionFn STRegion-1adb) (increasing LeftRightPositionFn STRegion-1adb))

(meets-inverse (constant PositionFn STRegion-790e) (increasing ForwardBackwardPositionFn STRegion-790e))

(meets-inverse (constant PositionFn STRegion-790e) (increasing LeftRightPositionFn STRegion-790e))

(meets-inverse (constant PositionFn STRegion-8adb) (increasing ForwardBackwardPositionFn STRegion-8adb))

(meets-inverse (constant PositionFn STRegion-d7e1) (increasing DistanceFn hand))

(meets-inverse (constant PositionFn STRegion-d7e1) (increasing ForwardBackwardPositionFn STRegion-d7e1))

(meets-inverse (constant PositionFn STRegion-d7e1) (increasing VolumeFn STRegion-d7e1))

(overlapped-by (changing PositionFn STRegion-07f4) (changing PositionFn hand))

(overlapped-by (changing PositionFn STRegion-1adb) (changing PositionFn hand))

(overlapped-by (changing PositionFn STRegion-6d01) (changing PositionFn hand))

(overlapped-by (changing PositionFn STRegion-790e) (changing PositionFn hand))

(overlapped-by (changing PositionFn STRegion-938a) (changing PositionFn hand))

(overlapped-by (changing PositionFn STRegion-d7e1) (decreasing DistanceFn hand))

(overlapped-by (changing PositionFn STRegion-d7e1) (holdsIn po hand STRegion-d7e1))

(overlapped-by (constant PositionFn STRegion-938a) (increasing DistanceFn hand))

(overlapped-by (constant PositionFn STRegion-938a) (increasing ForwardBackwardPositionFn STRegion-938a))

(overlapped-by (constant PositionFn STRegion-938a) (increasing LeftRightPositionFn STRegion-938a))

(overlaps (changing PositionFn STRegion-07f4) (constant PositionFn hand))

(overlaps (changing PositionFn STRegion-1adb) (constant PositionFn hand))

(overlaps (changing PositionFn STRegion-1adb) (holdsIn dc hand STRegion-1adb))

(overlaps (changing PositionFn STRegion-6d01) (constant PositionFn hand))

(overlaps (changing PositionFn STRegion-790e) (constant PositionFn hand))

(overlaps (changing PositionFn STRegion-790e) (holdsIn dc hand STRegion-790e))

(overlaps (changing PositionFn STRegion-938a) (constant PositionFn hand))

(overlaps (changing PositionFn STRegion-938a) (holdsIn dc hand STRegion-938a))

(overlaps (changing PositionFn STRegion-d7e1) (constant PositionFn hand))

(overlaps (changing PositionFn STRegion-d7e1) (holdsIn tpp hand STRegion-d7e1))

(overlaps (changing PositionFn STRegion-d7e1) (increasing DistanceFn hand))

(overlaps (constant PositionFn STRegion-d7e1) (decreasing DistanceFn hand))

(overlaps (constant PositionFn STRegion-d7e1) (holdsIn po hand STRegion-d7e1))

(overlaps (constant PositionFn STRegion-d7e1) (increasing ForwardBackwardPositionFn STRegion-d7e1))

(overlaps (constant PositionFn STRegion-d7e1) (increasing VolumeFn STRegion-d7e1))

(started-by (changing PositionFn STRegion-1adb) (decreasing LeftRightPositionFn STRegion-1adb))

(started-by (changing PositionFn STRegion-1adb) (holdsIn ec hand STRegion-1adb))

(started-by (changing PositionFn STRegion-790e) (holdsIn ec hand STRegion-790e))

(started-by (changing PositionFn STRegion-938a) (holdsIn ec hand STRegion-938a))

(started-by (changing PositionFn STRegion-d7e1) (changing PositionFn hand))

(started-by (constant PositionFn STRegion-d7e1) (constant DistanceFn hand))

(started-by (constant PositionFn STRegion-d7e1) (constant ForwardBackwardPositionFn STRegion-d7e1))

(started-by (constant PositionFn STRegion-d7e1) (constant PositionFn hand))

(started-by (constant PositionFn STRegion-d7e1) (constant VolumeFn STRegion-d7e1))

(started-by (constant PositionFn STRegion-d7e1) (holdsIn ec hand STRegion-d7e1))

(starts (changing PositionFn STRegion-1adb) (constant HeightFn STRegion-1adb))

(starts (changing PositionFn STRegion-1adb) (constant VolumeFn STRegion-1adb))

(starts (changing PositionFn STRegion-790e) (constant HeightFn STRegion-790e))

(starts (changing PositionFn STRegion-790e) (constant VolumeFn STRegion-790e))

(starts (changing PositionFn STRegion-790e) (increasing DistanceFn hand))

(starts (changing PositionFn STRegion-81e4) (constant ForwardBackwardPositionFn STRegion-81e4))

(starts (changing PositionFn STRegion-81e4) (constant HeightFn STRegion-81e4))

(starts (changing PositionFn STRegion-81e4) (constant LeftRightPositionFn STRegion-81e4))

(starts (changing PositionFn STRegion-81e4) (constant VolumeFn STRegion-81e4))

(starts (changing PositionFn STRegion-81e4) (holdsIn po hand STRegion-81e4))

(starts (changing PositionFn STRegion-81e4) (increasing DistanceFn hand))

(starts (changing PositionFn STRegion-8adb) (constant HeightFn STRegion-8adb))

(starts (changing PositionFn STRegion-8adb) (constant LeftRightPositionFn STRegion-8adb))

(starts (changing PositionFn STRegion-8adb) (constant VolumeFn STRegion-8adb))

(starts (changing PositionFn STRegion-8adb) (holdsIn po hand STRegion-8adb))

(starts (changing PositionFn STRegion-8adb) (increasing DistanceFn hand))

(starts (changing PositionFn STRegion-938a) (constant HeightFn STRegion-938a))

(starts (changing PositionFn STRegion-938a) (constant VolumeFn STRegion-938a))

(starts (changing PositionFn STRegion-938a) (increasing DistanceFn hand))

(starts (changing PositionFn STRegion-938a) (increasing ForwardBackwardPositionFn STRegion-938a))

(starts (changing PositionFn STRegion-938a) (increasing LeftRightPositionFn STRegion-938a))

(starts (constant PositionFn STRegion-ab5f) (constant ForwardBackwardPositionFn STRegion-ab5f))

(starts (constant PositionFn STRegion-ab5f) (constant HeightFn STRegion-ab5f))

(starts (constant PositionFn STRegion-ab5f) (constant VolumeFn STRegion-ab5f))

(starts (constant PositionFn STRegion-ab5f) (holdsIn ec hand STRegion-ab5f))

(starts (constant PositionFn STRegion-ab5f) (constant DistanceFn hand))

(starts (constant PositionFn STRegion-d7e1) (constant HeightFn STRegion-d7e1))

(starts (constant PositionFn STRegion-d7e1) (constant LeftRightPositionFn STRegion-d7e1))

(starts (constant PositionFn hand) PARENT_EPISODE)

(temporally-contains (constant PositionFn STRegion-d7e1) (changing PositionFn hand))

(temporally-equals (changing PositionFn STRegion-07f4) (constant VolumeFn STRegion-07f4))

(temporally-equals (changing PositionFn STRegion-07f4) (decreasing HeightFn STRegion-07f4))

(temporally-equals (changing PositionFn STRegion-07f4) (holdsIn dc hand STRegion-07f4))

(temporally-equals (changing PositionFn STRegion-07f4) (increasing DistanceFn hand))

(temporally-equals (changing PositionFn STRegion-07f4) (increasing ForwardBackwardPositionFn STRegion-07f4))

(temporally-equals (changing PositionFn STRegion-07f4) (increasing LeftRightPositionFn STRegion-07f4))

(temporally-equals (changing PositionFn STRegion-1adb) (increasing DistanceFn hand))

(temporally-equals (changing PositionFn STRegion-1adb) (increasing ForwardBackwardPositionFn STRegion-1adb))

(temporally-equals (changing PositionFn STRegion-3499) (constant VolumeFn STRegion-3499))

(temporally-equals (changing PositionFn STRegion-3499) (decreasing LeftRightPositionFn STRegion-3499))

(temporally-equals (changing PositionFn STRegion-3499) (holdsIn dc hand STRegion-3499))

(temporally-equals (changing PositionFn STRegion-3499) (increasing DistanceFn hand))

(temporally-equals (changing PositionFn STRegion-3499) (increasing ForwardBackwardPositionFn STRegion-3499))

(temporally-equals (changing PositionFn STRegion-3499) (increasing HeightFn STRegion-3499))

(temporally-equals (changing PositionFn STRegion-6d01) (constant VolumeFn STRegion-6d01))

(temporally-equals (changing PositionFn STRegion-6d01) (decreasing HeightFn STRegion-6d01))

(temporally-equals (changing PositionFn STRegion-6d01) (holdsIn dc hand STRegion-6d01))

(temporally-equals (changing PositionFn STRegion-6d01) (increasing DistanceFn hand))

(temporally-equals (changing PositionFn STRegion-6d01) (increasing ForwardBackwardPositionFn STRegion-6d01))

(temporally-equals (changing PositionFn STRegion-6d01) (increasing LeftRightPositionFn STRegion-6d01))

(temporally-equals (changing PositionFn STRegion-790e) (increasing ForwardBackwardPositionFn STRegion-790e))

(temporally-equals (changing PositionFn STRegion-790e) (increasing LeftRightPositionFn STRegion-790e))

(temporally-equals (changing PositionFn STRegion-8adb) (increasing ForwardBackwardPositionFn STRegion-8adb))

(temporally-equals (changing PositionFn STRegion-938a) (decreasing HeightFn STRegion-938a))

(temporally-equals (changing PositionFn STRegion-938a) (increasing DistanceFn hand))

(temporally-equals (changing PositionFn STRegion-938a) (increasing ForwardBackwardPositionFn STRegion-938a))

(temporally-equals (changing PositionFn STRegion-938a) (increasing LeftRightPositionFn STRegion-938a))

(temporally-equals (changing PositionFn STRegion-ab5f) (increasing LeftRightPositionFn STRegion-ab5f))

(temporally-equals (constant PositionFn STRegion-1adb) (constant DistanceFn hand))

(temporally-equals (constant PositionFn STRegion-1adb) (constant ForwardBackwardPositionFn STRegion-1adb))

(temporally-equals (constant PositionFn STRegion-1adb) (constant LeftRightPositionFn STRegion-1adb))

(temporally-equals (constant PositionFn STRegion-790e) (constant ForwardBackwardPositionFn STRegion-790e))

(temporally-equals (constant PositionFn STRegion-790e) (constant LeftRightPositionFn STRegion-790e))

(temporally-equals (constant PositionFn STRegion-8adb) (constant ForwardBackwardPositionFn STRegion-8adb))

(temporally-equals (constant PositionFn STRegion-8bbc) (constant DistanceFn hand))

(temporally-equals (constant PositionFn STRegion-8bbc) (constant ForwardBackwardPositionFn STRegion-8bbc))

(temporally-equals (constant PositionFn STRegion-8bbc) (constant HeightFn STRegion-8bbc))

(temporally-equals (constant PositionFn STRegion-8bbc) (constant LeftRightPositionFn STRegion-8bbc))

(temporally-equals (constant PositionFn STRegion-8bbc) (constant VolumeFn STRegion-8bbc))

(temporally-equals (constant PositionFn STRegion-8bbc) (holdsIn dc hand STRegion-8bbc))

(temporally-equals (constant PositionFn STRegion-ab5f) (constant LeftRightPositionFn STRegion-ab5f))

(temporally-equals (constant PositionFn STRegion-d7e1) (constant DistanceFn hand))

(temporally-equals (constant PositionFn STRegion-d7e1) (constant ForwardBackwardPositionFn STRegion-d7e1))

(temporally-equals (constant PositionFn STRegion-d7e1) (constant VolumeFn STRegion-d7e1))
(isa AILEEN CognitiveRobot)