A Task Algebra For Agents In Reinforcement Learning

Geraud Nangue Tasse
2291200

Supervisors:
Benjamin Rosman
Steven James

WITS
UNIVERSITY

A thesis submitted to the Faculty of Science, University of Witwatersrand, in fulfilment of the requirements for the degree of Master of Science.

January 2020

Abstract

A necessary property for generally intelligent or lifelong-learning agents is the ability to reuse the knowledge learned from old tasks to solve new tasks. This knowledge reuse can come in the form of zero-shot learning—where it is sufficient to immediately solve new tasks—or few-shot learning—where some additional learning needs to be done to solve new tasks. Of particular interest is the class of tasks that can solved via zero-learning since it leads to a direct way of generalising over a problem space. One such class of tasks seems to be tasks specified by the arbitrary logical composition of already solved tasks. That is tasks specified by arbitrary disjunction (union), conjunction (intersection), and complement (negation) of learned tasks. The potential for zero-shot learning here stems from the intuitive understanding that humans seem to have of the union, intersection, and negation of tasks that they know. This zero-shot learning problem is yet to be solved despite the general success of reinforcement learning in the past decade and the current success of transfer learning methods that involve policies or value functions composition. One possible cause of this is that there is no unifying formalism for the disjunction, conjunction, and negation of tasks.

This work addresses the problem first by formally defining the composition of tasks as operators acting on a set of tasks in an algebraic structure. This provides a structured way of doing task compositions and a theoretically rigorous way of studying them. We propose a framework for defining lattice algebras and Boolean algebras in particular over the space of tasks. This allows us to formulate new tasks in terms of the negation, disjunction, and conjunction of a set of base tasks. We then show that by learning a new type of goal-oriented value functions and restricting the rewards of the tasks, an agent can solve composite tasks with no further learning.

We verify our approach in two domains—including a high-dimensional video game environment requiring function approximation—where an agent first learns a set of base skills, and then composes them to solve a super-exponential number of new tasks.

Acknowledgements

I would like to thank my supervisor Benjamin Rosman and co-supervisor Steven James for their amazing support and advices throughout this project. I extend my gratitude to all those who directly or indirectly supported and helped me during this journey. Above all I am deeply grateful to my family and friends for their emotional support and encouragements.

Declaration

I, Geraud Nangue Tasse, declare that this thesis is my own, unaided work. It is being submitted for the Degree of Masters of Science at the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination at any other University.

[illegible]

(Signature of candidate)

03 day of August 20 20 in Pretoria

to my family

Contents

Chapter 1

Introduction

1.1 Overview

Reinforcement learning (RL) has achieved recent success in a number of difficult, high-dimensional environments (Mnih *et al.*, 2015; Levine *et al.*, 2016; Lillicrap *et al.*, 2016; Silver *et al.*, 2017). However, these methods generally require millions of samples from the environment to learn optimal behaviours, limiting their real-world applicability. A major challenge is thus in designing sample-efficient agents that can transfer their existing knowledge to solve new tasks quickly. This is particularly important for agents in a multitask or lifelong setting, since learning to solve complex tasks from scratch is typically infeasible.

Lifelong learning poses the problem of learning in the context in which an agent is presented with a number of tasks during it's lifetime, and needs to be able to transfer knowledge learned from previous tasks to solve new tasks (Abel *et al.*, 2018; Thrun, 1996). It is motivated by how humans are seemingly able to generalize over large space of problems from solving a small subset of it. Building lifelong learning agents is one of the ultimate goals in reinforcement learning, as having such agents will make them more practical in many real life problems. One approach towards achieving it is *composition* (Todorov, 2009), which allows an agent to leverage existing skills to build complex, novel behaviours. These newly-formed skills can then be used to immediately solve new tasks (zero-shot composition) or speed up the learning of new tasks (few-shot composition). Zero-shot composition is of particular interest to the lifelong learning problem, as it gives a direct way for agents to quickly generalize over a task space.

A desirable trait of agents is the ability to do zero-shot composition on any class of tasks for which past knowledge is sufficient to solve knew tasks. A candidate for such tasks are the logical combinations of previously solved tasks. That is, new tasks formed by using the logic operators OR, AND, and NOT on past tasks. This type of task is of great interest to the lifelong learning problem since humans are able to solve them seemingly with great ease.

Consider for example a domain with tasks corresponding to collecting various objects of possibly different shapes and colors (Figure 1.1). A human that has learned how to collect any squares optimally—*Square* task—and separately learned how to collect any blue object optimally—*Blue* task—can immediately solve any of the following tasks,

- *Square* **OR** *Blue*: Collect squares or blue objects optimally.
- *Square* **AND** *Blue*: Collect squares that are blue optimally.

- **NOT** *Square*: Collect objects that are not squares optimally.
- **NOT** *Blue*: Collect objects that are not blue optimally.
- *Square* **AND** (**NOT** *Blue*): Collect squares that are not blue optimally.
- *Blue* **AND** (**NOT** *Crate*): Collect blue objects that are not squares optimally.
- (*Square* **OR** *Blue*) **AND** **NOT**(*Square* **AND** *Blue*): Collect squares or blue objects that are not blue squares optimally.

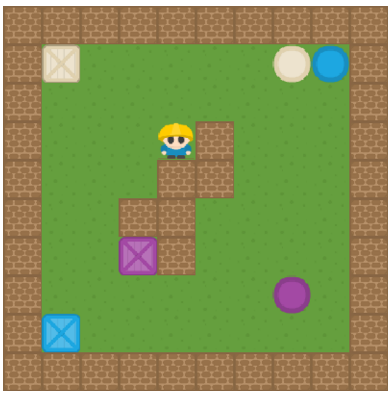


Figure 1.1: Video game domain with different composable tasks (Van Niekerk *et al.*, 2019)

This seems to be because as humans we are able to do the disjunction (union), conjunction (intersection), and negation (opposite) of tasks we know (given some assumptions about said tasks). Our ability to immediately solve logical combinations of known tasks also suggests that our knowledge representation for learned tasks is sufficiently general. That is, it is not limited to the specific objectives of the task. This makes sense since during our experience learning tasks we do not learn only about the best outcomes, but also about the less valuable or even bad ones (White, 1959). This generality in knowledge and compositional ability gives us a very simple degree of compositional explosion of skills. It is hence much desired for lifelong agents to have the same ability. As an agent learns new tasks it should gain sufficient knowledge from them to understand how they can be combined together to solve new ones. That way whenever it is desired for the agent to solve a logical combination of previous tasks, it can simply proceed. This is essential for using agents practically since for each new desired task like *collect squares but not blue ones*, one would not want to wait for the agent to learn how to do it when it is seemingly trivial to us.

In this work we are hence interested in the logical combination of tasks and skills. Since logics are abstracted in lattice theory (Birkhoff, 1940), we use the algebraic structures therein to formalise the logical composition of tasks and skills in reinforcement learning. We first define a lattice algebra over the set of tasks. This formalises and unifies the disjunctive and conjunctive compositions considered by previous works (Todorov, 2009; Saxe *et al.*, 2017; Haarnoja *et al.*, 2018; Van Niekerk *et al.*, 2019; Hunt *et al.*, 2019; Peng *et al.*, 2019). We then give a formal meaning to the negation of tasks, which is used to define a De Morgan algebra over the task space. This extends previous composition problems to encompass all logic operators: conjunction, disjunction, and negation. We show that for tasks with sparse rewards, the De Morgan algebra extends to a Boolean algebra. This gives us the notion of base tasks.

We introduce a new type of value function called *extended value function* which encodes how to achieve various outcomes for a single task. The same algebraic structures defined for the task space are also defined for a value function space. We then use the rich knowledge encoded by the extended value functions to prove zero-shot composition. This is done by showing that there exists a homomorphism between the task and value function algebras. Given any new task specified as the composition of learned tasks, we can immediately obtain the value function that solves it. Furthermore, given a set of base tasks that have been previously solved by the agent, any new task written as a Boolean expression can immediately be solved without further learning, resulting in a zero-shot super-exponential explosion in the agent's abilities.

We illustrate our approach in simple gridworld domains, where an agent first learns to go to particular regions of the xy-plane, after which it can then optimally solve any task specified as their logical combination. We then demonstrate composition in high-dimensional video game environments, where an agent first learns to collect different objects, and then compose these abilities to solve complex tasks immediately. Our results show that, even when function approximation is required, an agent can leverage its existing skills to solve new tasks without further learning.

1.2 Research Problem

In Section 1.1 we discussed lifelong learning agents, and how humans have various transfer learning abilities which are much desired for these agents. We discussed one such ability, which is how humans are able to immediately solve new tasks posed to them as logical combinations of previously solved tasks. This is one way in which humans are able to generalise over the space of problems. Hence it is necessary for agents to have this ability for the ultimate goal of lifelong learning agents. This is a problem of great significance that has yet to be solved.

In this work we break down the problem into three parts. First is how can we formalise what is meant by the logical combination of tasks in the context of reinforcement learning. Consider for example the video game domain shown in Figure 1.1. If *Square* and *Blue* are tasks with formal definitions, then the question is: What is the formal task defined by logical statements like *Square* **AND** *Blue* or even (*Square* **OR** *Blue*) **AND** **NOT**(*Square* **AND** *Blue*)? This is important because while we know intuitively what such specifications mean, we do not have rigorous definitions for them. We hence cannot begin to study how to optimally solve compositional tasks when their definition is still lacking.

The second part of the problem is what type of knowledge do agents need to learn about previous tasks to have any chance of solving new tasks optimally without learning. This is an important problem as it is not necessarily true that the usual knowledge representation in reinforcement learning is sufficient to solve compositional tasks. We can already see how for the simple example of *Square* **AND** *Blue* task, an agent needs to know about all the squares and circles in the environment, be able to analyze them to see which ones satisfy both properties, and be able to compare those to see which one is the nearest (in the case where there is more than one).

Finally we ask what method can be used to combine the knowledge learned from previous tasks to solve new tasks optimally. This is the main problem we aim to address.

1.3 Main Contributions

We describe the main contributions of this work.

1. *Task algebra (Chapter 3):* We formalise the disjunction, conjunction, and negation of tasks as operators acting on a set of tasks in lattice structures. This introduces a notion of task space, and enables interpretable and easy task specifications. This is important in complex lifelong settings where constructing rewards for new tasks is hard. A main result obtained from this formalism is the notion of base tasks, which are a minimal set of tasks that can be determined and which are sufficient to specify any other task in a task space. This is essential in the lifelong setting where the task space is simply too large to construct rewards for all desired tasks.

2. *Extended value functions (Chapter 4):* This is a new type of value function which we introduce to later achieve zero-shot composition. This value function is a goal-oriented value function that encodes how to achieve various goals in an environment. It is important for transfer learning in general as it is learned from a single task but encodes richer knowledge than standard value functions.

3. *Zero-shot composition (Chapter 5):* We prove that any task specified as the logical composition of learned tasks can be solved immediately without further learning. We demonstrate via a series of experiments that this holds true not only theoretically but also practically, even when the assumptions made in the theory are broken. This is an important result as it enables lifelong learning agents to solve a super-exponentially increasing number of tasks as the number of base tasks they learn increase.

1.4 Thesis Structure

Since we are interested in solving tasks in the context of reinforcement learning and formalising their logical composition using lattice theory, we give a brief introduction to these areas in Chapter 2. We define and describe the main concepts and structures that will be used throughout the thesis. In particular Chapters 3 and 5 formalises the logical composition of tasks and value functions respectively using the lattice structures. They hence require familiarity with the definitions and main properties of these structures, all of which is stated in Chapter 2.

Chapter 3 uses the structures described in Chapter 2 to formalise the disjunction, conjunction, and negation of tasks. This gives us a notion of task space. The set of tasks we consider are a restricted class of Markov decision processes that model goal reaching tasks. Hence the logical operators we define act on Markov decision processes such that composing them produces new Markov decision processes. With this task algebra established, we leverage the mathematical richness that comes with it to show that for tasks with sparse rewards, there is a relationship between the task space and the power set of the goal space. This leads to the formal introduction of base tasks, which has been so far mentioned informally in the literature. We finally show that standard value functions are insufficient to solve logical task compositions without extra learning.

Chapter 4 develops a new goal-oriented value function that encodes more information about solved tasks than standard value functions. It establishes the main theory for it by following a similar structure to that of value functions given in Chapter 2. Main properties of these value functions are also stated and proven here. Some of these properties are heavily relied on in Chapter 5 to establish the relevant algebraic structures for value functions and show zero-shot composition. In Chapter 4 we also propose an algorithm for learning the new value function and conduct relevant experiments on it. Mainly, we demonstrate that these value functions do indeed encode a variety of skills learned from the same type of experience as standard value functions. In particular they encode how to achieve various goals in the environment.

Chapter 5 formalises the disjunction, conjunction, and negation of value functions in the algebraic structures of lattices, similarly to how the task algebra was established. This gives us a notion of value function space. The intuition here is that to successfully achieve zero-shot composition, we want the notion of logics in the space of knowledge learned (value functions) to be similar to that in the space of tasks they were learned from. We in fact prove zero-shot composition here by showing that the task space and value function space are homomorphic. The zero-shot compositions are first demonstrated in gridworld domains where value functions can be learned optimally. We then demonstrate that these compositions still hold in high dimensional environments, where function approximation is required. The high dimensional environment considered here is the video game domain illustrated in Section 1.1, as we show how the logical compositions we desired from lifelong agents are now achieved. Finally we investigate some practical considerations here. In particular we investigate the effect of dropping the assumptions made for theoretical guarantees. We demonstrate that zero-shot composition still holds thanks to the rich knowledge represented by the introduced value functions.

Chapter 2

Preliminaries

2.1 Introduction

In this Chapter we describe the main concepts and algebraic structures that will be used throughout this thesis. Section 2.2 gives a brief introduction to reinforcement learning, describing how decision problems are modelled and some important concepts like policies and value functions relevant for later chapters. In Section 2.3 we also briefly introduce lattice theory and describe the main structures of interest.

2.2 Reinforcement Learning

In its simplest form, reinforcement learning is about agents learning to make decisions so as to maximize some numerical quantity. Given a decision-problem, the learner determines the best decisions to make at each point in time by repeated trial and error. These decision-problems are usually formulated as Markov decision processes (MDPs) (Puterman, 2014). The MDP encodes information about the possible states the environment can be in, the possible decisions (actions) the agent can make, the behaviour (transition dynamics) of the environment in response to the decisions the learner makes, and the numerical quantity (rewards) that the learner receives for each decision it makes.

The reward signal encodes the objective of a reinforcement learning problem. An agent's objective is usually to take actions that maximize the total rewards it expects to receive over the future. This is encapsulated by the reward hypothesis:

All of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (called reward). (Sutton *et al.*, 1998)

This notion of an agent's objective may at first appear limiting but has proven to be sufficient for a wide range of problems. These range from agents learning how to play games at an expert level (Mnih *et al.*, 2013; Fu, 2016) to controlling simulated and physical robots (Peng *et al.*, 2018).

The function that determines what action the agent takes at a given state is called its *policy* and the function that gives the total rewards it expects to receive over the future following that policy is called the *value function*.

2.2.1 Markov Decision Processes

MDPs formalise the class of problems involving an agent interacting with an environment to reach some goals (Figure 2.1) . A state $s = \mathcal{S}_t$ of the environment is said to be *Markov* iff,

$$\mathbb{P}[\mathcal{S}_{t+1} = s' | \mathcal{S}_t = s, \mathcal{A}_t = a] = \mathbb{P}[\mathcal{S}_{t+1} = s' | \mathcal{S}_0 = s, \mathcal{A}_0 = a, ..., \mathcal{S}_t = s, \mathcal{A}_t = a].$$

A decision-making problem that satisfies the Markov property $\forall s \in \mathcal{S}$ is called a Markov decision process (MDP). Formally, an MDP is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$ where,

- $\mathcal{S}$ is the state space,
- $\mathcal{A}$ is the action space,
- p is a Markov transition function,

$$\begin{aligned} p \colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} &\to [0,1] \\ (s, a, s') &\mapsto \mathbb{P}[\mathcal{S}_{t+1} = s' | \mathcal{S}_t = s, \mathcal{A}_t = a], \end{aligned}$$

- r is the reward function, $r \colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$,
- and $\gamma \in [0,1]$ is the discount factor

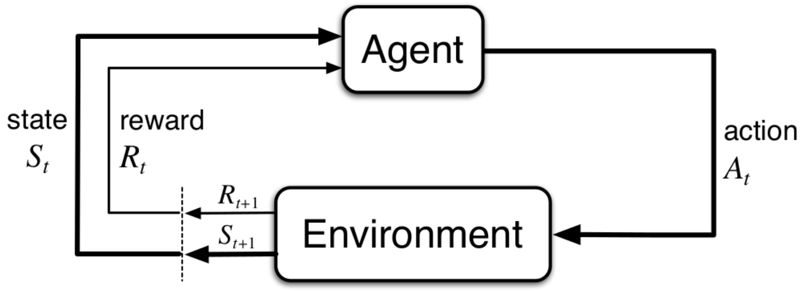


Figure 2.1: Illustration of agent-environment interaction (Sutton *et al.*, 1998)

An agent's objective is usually to maximise the return G_t for these MDPs,

$$G_t \coloneqq R_{t+1} + \gamma R_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}. \tag{2.1}$$

Note how G_t diverges if $\gamma = 1$ and the rewards are non-zero. Usually in this case the state space is augmented with a special state, w, such that $p(s, a, w) = 1 \; \forall (s, a) \in (\mathcal{G} \times \mathcal{A})$ and the rewards are zero after reaching w. These are undiscounted MDPs where an agent needs to reach some goal states in $\mathcal{G} \subset \mathcal{S}$.

MDPs with finite $\mathcal{S}$ and $\mathcal{A}$ are called finite MDPs, otherwise they are called infinite MDPs. We consider finite MDPs throughout this work; however, all the results also hold for infinite MDPs except where otherwise stated.

2.2.2 Policies and value functions

Formally, a Markov policy π is defined as the probability of selecting each action in a given state. That is,

$$\begin{aligned} \pi \colon \mathcal{A} \times \mathcal{S} &\to [0,1] \\ (a,s) &\mapsto \mathbb{P}[\mathcal{A}_t = a | \mathcal{S}_t = s]. \end{aligned}$$

The state-value function v under a policy π is defined as the expected return of starting at a given state and following π thereafter. That is,

$$\begin{aligned} V^{\pi} \colon \mathcal{S} &\to \mathbb{R} \\ s &\mapsto \mathbb{E}_{\pi}[G_t | \mathcal{S}_t = s]. \end{aligned}$$

Similarly, the action-value function q under a policy π is defined as,

$$\begin{aligned} Q^{\pi} \colon \mathcal{S} \times \mathcal{A} &\to \mathbb{R} \\ (s,a) &\mapsto \mathbb{E}[G_t | \mathcal{S}_t = s, \mathcal{A}_t = a]. \end{aligned}$$

The action-value function is particularly useful in model-free reinforcement since it enables agents to learn optimal actions for each state without the need for transition dynamics.

Using the recursive property of G_t, the value functions can be shown to satisfy the following Bellman equations (Bellman, 1954),

$$V^{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(a,s) \left(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s,a,s') V^{\pi}(s') \right), \tag{2.2}$$

$$Q^{\pi}(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s,a,s') \sum_{a \in \mathcal{A}} \pi(a,s) Q^{\pi}(s',a). \tag{2.3}$$

By defining a partial ordering over policies, it can be shown that $\exists \pi^* \geq \pi, \forall \pi$, which leads to the optimal value functions,

$$V^{\pi^*}(s) = V^*(s) = \max_{\pi} V^{\pi}(s) \ \ \forall \pi^*,$$

$$Q^{\pi^*}(s,a) = Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a) \ \ \forall \pi^*.$$

These give rise to the following Bellman optimality equations for value functions,

$$V^*(s) = \max_{a \in \mathcal{A}} \left(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s,a,s') V^*(s') \right), \tag{2.4}$$

$$Q^*(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s,a,s') \max_{a \in \mathcal{A}} Q^*(s',a). \tag{2.5}$$

These are written more succinctly using the Bellman operators as follows,

$$\left[\mathcal{T}^{\bar{\pi}} V\right](s) = r(s,\pi(s)) + \gamma \sum_{s' \in \mathcal{S}} p(s,\pi(s),s') V(s'), \tag{2.6}$$

$$[\mathcal{T}V](s) = \max_{a \in \mathcal{A}} \left[r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s,a,s')V(s') \right]. \tag{2.7}$$

An optimal policy π^* can also be obtained by acting greedily on the optimal action-value function Q^*. That is,

$$\pi^*(a,s) = \begin{cases} 1 & a = \arg\max_{a \in \mathcal{A}} Q^*(s,a), \\ 0 & \text{otherwise.} \end{cases} \tag{2.8}$$

2.3 Lattice Theory

Lattice theory is a generalization of the study of Boolean algebras (Grätzer, 2002). Boolean algebra is today widely applied to various areas such as propositional logics, set theory, logic gates in electronics and much more. In general it is the go-to framework for formalising concepts that have a logical nature. Its formulation dates back to Boole (1854) in his seminal work on *the laws of thought*, who developed the precursor to current Boolean algebra while attempting to formalize propositional logics. The notion of a lattice was later introduced in the field as a useful generalisation of the axioms of Boolean algebra. Indeed lattice theory is now pervasive through the whole of modern algebra (Birkhoff, 1940).

Lattice theory can be studied as part of order theory or abstract algebra. The basic concepts in the order theoretic view are: Partial orders, least upper bounds and greatest lower bounds. These are equivalent to its basic concepts in the algebraic view: Join (abstracts disjunction) and meet (abstracts conjunction) semi-lattices. This simplicity yet richness in applications is the main appeal of lattice theory. Since we are mainly interested in the conjunction, disjunction, and negation operators, our main focus will be on the algebraic formulation as we build our way up to a Boolean algebra. However we will also use the order theoretic view whenever that seems most natural. We describe here the main structures that are of interest to us.

2.3.1 Partial-order

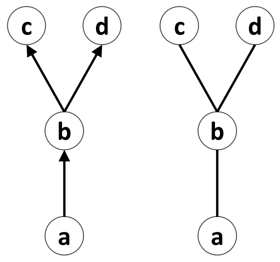
Definition 2.3.1. *A partially ordered set (poset) $(\mathcal{L}, \leq)$ is a set $\mathcal{L}$ equipped with the binary relation $\leq$ (less than or equal to) which satisfies the following properties for a, b, c in $\mathcal{L}$:*

(i) Reflexivity: $a \leq a$.

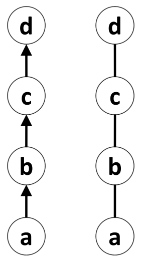
(ii) Antisymmetry: $if a \leq b$ and $b \leq a$, then $a = b$.

(iii) Transitivity: $if a \leq b$ and $b \leq c$, then $a \leq c$.

The relation $\leq$ is called the *partial order* of $\mathcal{L}$. A special type of partial order is a well order. A well ordered set is a poset that has a least element and for which every element is comparable. Figure 2.2 illustrates a poset and a well ordered set using a Hasse diagram, which will be used throughout.



(a) Single rooted tree



(b) Chain of a,b,c,d.

Figure 2.2: Example of a partial order and a well order. (a) Shows a partial order drawn as a directed graph and as a Hasse diagram. (b) Shows a well order also drawn as a directed graph and as a Hasse diagram. The Hasse diagram is drawn such that if $a \leq b$, then $a \in \mathcal{L}$ is drawn below $b \in \mathcal{L}$. An edge is drawn between a and b if they are comparable and one covers the other. That is if say $a \leq b$ and they are immediate neighbours.

2.3.2 Semi-Lattice Algebra

Join Semi-Lattice

In order theory, a join semi-lattice is a partial order $(\mathcal{L}, \leq)$ in which every pair of elements $a, b \in \mathcal{L}$ has a *least upper bound (their supremum)* (Grätzer, 2011). The algebraic definition is as follows:

Definition 2.3.2. *A join semi-lattice is a set $\mathcal{L}$ equipped with the binary operator $\vee$ (join, disjunction) which satisfies the following join semi-lattice axioms for a, b, c in $\mathcal{L}$:*

(i) Idempotence: $a \vee a = a$.

(ii) Commutativity: $a \vee b = b \vee a$.

(iii) Associativity: $a \vee (b \vee c) = (a \vee b) \vee c$.

A join semi-lattice algebra $(\mathcal{L}, \vee)$ induces a join semi-lattice order $(\mathcal{L}, \leq)$ with the relation $\leq$ given by

$$a \leq b \iff a \vee b = b.$$

Similarly, a join semi-lattice order $(\mathcal{L}, \leq)$ induces a join semi-lattice algebra $(\mathcal{L}, \vee)$ with the binary operator $\vee$ given by

$$a \vee b = \sup\{a, b\}.$$

Note that L is bounded above if

$$\bigvee \mathcal{L} = \sup_{L \in \mathcal{L}} L$$

exists. Hence the order theoretic and the algebraic definitions are equivalent. We will henceforth be using the algebraic definition, noting the induced partial order whenever needed.

Example 2.3.1. *Consider a finite set $\mathcal{L}$ with cardinality $|\mathcal{L}| \geq 1$. Then the set $\mathcal{M}$ of subsets $\mathcal{N} \subseteq \mathcal{L}$ with cardinalities $|\mathcal{N}| \geq \lceil \frac{|\mathcal{L}|}{2} \rceil$ forms a partial order under the usual $\subseteq$ relation. Hence it also forms a join semi-lattice $(\mathcal{M}, \vee)$ with*

$$a \vee b = \sup\{a, b\} \; \forall a, b \in \mathcal{M}.$$

Figure 2.4 shows the Hasse diagram of such a semi-lattice for a 4 element set, $\{a, b, c, d\}$.

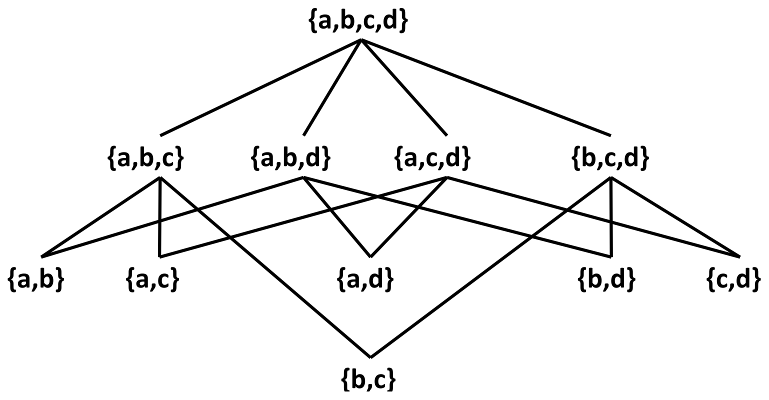


Figure 2.3: Join semi-lattice on subsets of $\{a, b, c, d\}$ with at least 2 elements. The Hasse diagram is ordered by *is a subset of.*

Meet Semi-Lattice

In order theory, a meet semi-lattice is a partial order $(\mathcal{L}, \leq)$ in which every pair of elements $a, b \in \mathcal{L}$ has a *greatest lower bound (their infimum)* (Grätzer, 2011). The algebraic definition is as follows:

Definition 2.3.3. *A meet semi-lattice $(\mathcal{L}, \wedge)$ is a set $\mathcal{L}$ equipped with the binary operator $\wedge$ (meet, conjunction) which satisfies the following meet semi-lattice axioms for a, b, c in $\mathcal{L}$:*

(i) Idempotence: $a \wedge a = a$.

(ii) Commutativity: $a \wedge b = b \wedge a$.

(iii) Associativity: $a \wedge (b \wedge c) = (a \wedge b) \wedge c$.

A meet semi-lattice algebra $(\mathcal{L}, \wedge)$ induces a meet semi-lattice order $(\mathcal{L}, \leq)$ with the relation $\leq$ given by

$$a \leq b \iff a \wedge b = a.$$

Similarly, a meet semi-lattice order $(\mathcal{L}, \leq)$ induces a meet semi-lattice algebra $(\mathcal{L}, \wedge)$ with the binary operator $\wedge$ given by

$$a \wedge b = \inf\{a, b\}.$$

Note that L is bounded below if

$$\bigwedge \mathcal{L} = \inf_{L \in \mathcal{L}} L$$

exists. Hence the order theoretic and the algebraic definitions are equivalent. We will henceforth be using the algebraic definition, noting the induced partial order whenever needed.

Example 2.3.2. *Any single rooted tree $(\mathcal{T}, <)$ forms a meet semi-lattice $(\mathcal{T}, \wedge)$ with*

$$a \wedge b = \inf\{a, b\}.$$

Figure 2.4 shows an example of this for a set with 10 elements.

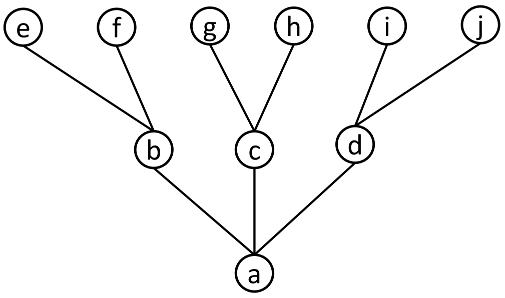


Figure 2.4: Meet semi-lattice induced by a tree with 10 nodes.

2.3.3 Lattice Algebra

In order theory, a lattice is a partial order $(\mathcal{L}, \leq)$ in which every pair of elements $a, b \in \mathcal{L}$ has a *least upper bound (their supremum)* and a *greatest lower bound (their infimum)* (Grätzer, 2011). The algebraic definition is as follows:

Definition 2.3.4. *A Lattice algebra $(\mathcal{L}, \vee, \wedge)$ is a set $\mathcal{L}$ equipped with the binary operators $\vee$ and $\wedge$ which satisfies the following lattice axioms for a, b, c in $\mathcal{L}$:*

(i) Idempotence: $a \wedge a = a \vee a = a$.

(ii) Commutativity: $a \wedge b = b \wedge a$ and $a \vee b = b \vee a$.

(iii) Associativity: $a \wedge (b \wedge c) = (a \wedge b) \wedge c$ and $a \wedge (b \vee c) = (a \vee b) \vee c$.

(iv) Absorption: $a \wedge (a \vee b) = a \vee (a \wedge b) = a$.

In other words a lattice is both a join semi-lattice and a meet semi-lattice that satisfies the absorption property.

A lattice algebra $(\mathcal{L}, \vee, \wedge)$ induces a lattice order $(\mathcal{L}, \leq)$ with the relation $\leq$ given by

$$a \leq b \iff a \vee b = b,$$

or

$$a \leq b \iff a \wedge b = a.$$

Both definitions of $\leq$ are the equivalent. The join is hence said to be the dual of meet. Similarly, a lattice order $(\mathcal{L}, \leq)$ induces a lattice algebra $(\mathcal{L}, \vee, \wedge)$ with the binary operators $\vee$ and $\wedge$ given by

$$a \vee b = \sup\{a, b\},$$

and

$$a \wedge b = \inf\{a, b\}.$$

Note that L is bounded if

$$\bigvee \mathcal{L} = \sup_{L \in \mathcal{L}} L,$$

and

$$\bigwedge \mathcal{L} = \inf_{L \in \mathcal{L}} L$$

exists. Hence the order theoretic and the algebraic definitions are equivalent. We will henceforth be using the algebraic definition, noting the induced partial order whenever needed.

Example 2.3.3. *Consider the partition of a set $\mathcal{L}$, which is a set of disjoint non-empty subsets of $\mathcal{L}$ such that its disjoint union gives back $\mathcal{L}$. The set of all partitions of any set $\mathcal{L}$ forms a lattice ordered by refinement. A partition refines another if all its elements are subsets of elements of the other partition (Comtet, 2012). Figure 2.5 shows the lattice structure of the partitions of a 4 element set.*

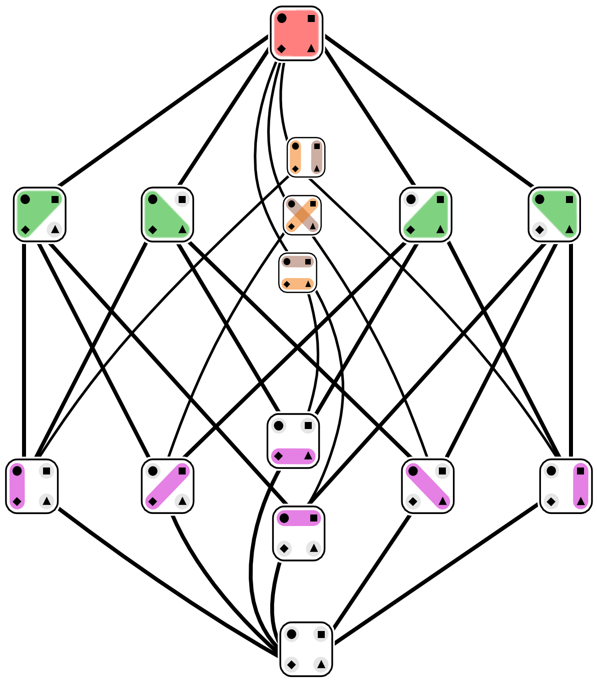


Figure 2.5: Hasse diagram of the partitions of a 4 element set.

2.3.4 De Morgan Algebra

The De Morgan algebra gives us an intuitive notion for negation, that is similar to that of propositional logics but with less restrictions (Monteiro, 1974).

Definition 2.3.5. *A De Morgan algebra $(\mathcal{L}, \vee, \wedge, \neg)$ is a set $\mathcal{L}$ equipped with the binary operators $\vee$ and $\wedge$, and the unary operator $\neg$ (involution,negation), which satisfies the following De Morgan algebra axioms for a, b, c in $\mathcal{L}$:*

(i) Idempotence: $a \wedge a = a \vee a = a$.

(ii) Commutativity: $a \wedge b = b \wedge a$ *and* $a \vee b = b \vee a$.

(iii) Associativity: $a \wedge (b \wedge c) = (a \wedge b) \wedge c$ *and* $a \wedge (b \vee c) = (a \vee b) \vee c$.

(iv) Absorption: $a \wedge (a \vee b) = a \vee (a \wedge b) = a$.

(v) Distributivity: $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ *and* $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$.

(vi) Identity: there exists $\mathbf{0}, \mathbf{1}$ *in* $\mathcal{L}$ *such that*

$$\begin{aligned} \mathbf{0} \wedge a &= \mathbf{0} \\ \mathbf{0} \vee a &= a \\ \mathbf{1} \wedge a &= a \\ \mathbf{1} \vee a &= \mathbf{1} \end{aligned}$$

(vii) De Morgan involution: $\neg\neg a = a$ *and* $\neg(a \vee b) = \neg a \wedge \neg b$.

In other words a De Morgan algebra is a bounded distributive lattice that's equipped with a De Morgan involution (a negation operator that satisfies the De Morgan laws).

Example 2.3.4. *De-Morgan algebra is the structure used for fuzzy logics (Klir & Yuan, 1995) where logic values range from 0 to 1. Fuzzy logics in turn is applied to a wide range of fields such as in control theory and artificial intelligence (Yen & Langari, 1999).*

2.3.5 Boolean Algebra

The Boolean algebra abstracts the concepts of propositional logics and set theory (Grätzer, 2011).

Definition 2.3.6. *A Boolean algebra* $(\mathcal{L}, \vee, \wedge, \neg)$ *is a set* $\mathcal{L}$ *equipped with the binary operators* $\vee$ *(disjunction) and* $\wedge$ *(conjunction), and the unary operator* $\neg$ *(negation), which satisfies the following Boolean algebra axioms for* a, b, c *in* $\mathcal{L}$*:*

(i) Idempotence: $a \wedge a = a \vee a = a$.

(ii) Commutativity: $a \wedge b = b \wedge a$ *and* $a \vee b = b \vee a$.

(iii) Associativity: $a \wedge (b \wedge c) = (a \wedge b) \wedge c$ *and* $a \wedge (b \vee c) = (a \vee b) \vee c$.

(iv) Absorption: $a \wedge (a \vee b) = a \vee (a \wedge b) = a$.

(v) Distributivity: $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ *and* $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$.

(vi) Identity: there exists $\mathbf{0}, \mathbf{1}$ *in* $\mathcal{L}$ *such that*

$$\begin{aligned} \mathbf{0} \wedge a &= \mathbf{0} \\ \mathbf{0} \vee a &= a \\ \mathbf{1} \wedge a &= a \\ \mathbf{1} \vee a &= \mathbf{1} \end{aligned}$$

(vii) *Complements: for every a in $\mathcal{L}$, there exists an element a' in $\mathcal{L}$ such that $a \wedge a' = \mathbf{0}$ and $a \vee a' = \mathbf{1}$.*

In other words a Boolean algebra is a De Morgan algebra whose involution satisfies the law of the excluded middle ($a \vee a' = \mathbf{1}$) and the law of non contradiction ($a \wedge a' = \mathbf{0}$).

This structure will be of particular interest as we aim to develop zero-shot logical composition of tasks.

2.4 Conclusion

Reinforcement learning deals with agents that learn to act optimally in an environment while lattice theory deals with abstracted concepts of logics. In this chapter we briefly introduced both fields and described some of their main concepts that will be used in later chapters.

Chapter 3

Composing Tasks

3.1 Introduction

A first step towards enabling agents to solve new compositional tasks is to formally define what these composed tasks are. Since in reinforcement learning tasks are commonly modelled as MDPs, we want principled ways of composing these MDPs to produce new MDPs that model a desired task specification. We do this by leveraging the structure of lattice algebras, since they are the mathematical structures that abstracts the notion of disjunction, conjunction and negation.

Hence the main contributions of this chapter are as follows:

- We establish the notion of a task lattice, which formalises the specification of tasks as arbitrary disjunctions and conjunctions of known tasks.
- We introduce a definition for the negation of tasks, allowing us to specify tasks as arbitrary disjunctions, conjunctions, and negations of known tasks. We formalise this under a De Morgan algebra.
- By adding the constraint that goals are either desirable or not, we formalise the full logical specification of tasks under a Boolean algebra.
- We show that this Boolean task algebra is in fact isomorphic to a power set algebra on the set of goals. This gives us the notion of base tasks that leads to a super-exponential explosion in task specifications.
- Finally we demonstrate that in general standard value functions do not encode enough knowledge about learned tasks to be able to solve logical task specifications without extra learning.

This chapter is structured as follows. We present related work on task and value function composition in Section 3.2. Since we intend to formalise the various compositions of tasks, we first describe the class of MDPs which will be considered as a set of tasks in Section 3.3. In Section 3.4 we formally establish composition of these tasks under relevant algebraic structures. Having formalised the logical specification of tasks, we show their relationship with the power set algebra in Section 3.5. With the tasks specified by composing other tasks formally defined, we show in Section 3.6 that standard value functions are insufficient to solve them without extra learning.

3.2 Related Work

The ability to compose tasks and value functions was first demonstrated using the linearly-solvable MDP framework (Todorov, 2007), where value functions could be composed to solve tasks similar to the disjunctive case (Todorov, 2009). Van Niekerk *et al.* (2019) show that the same kind of composition can be achieved using entropy-regularised RL (Fox *et al.*, 2016), and extend the results to the standard RL setting, where agents can optimally solve the disjunctive case. Using entropy-regularised RL, Haarnoja *et al.* (2018) approximates the conjunction of tasks by averaging their reward functions, and demonstrates that by averaging the optimal value functions of the respective tasks, the agent can achieve near-optimal performance. Hunt *et al.* (2019) extends this result by composing value functions to solve the average reward task exactly, which approximates the true conjunctive case. More recently, Peng *et al.* (2019) introduce a few-shot learning approach to compose policies multiplicatively. Although lacking theoretical foundations, results show that an agent can learn a weighted composition of existing base skills to solve a new complex task.

All these works consider disjunctions and conjunctions of tasks separately. In this chapter we unify them to obtain tasks specified as arbitrary disjunction of conjunctions (or conjunction of disjunctions), and further extend them to include negations.

3.3 Tasks

In this work we are interested in goal-reaching tasks. This is because they lend themselves to the lifelong setting where an agent is given tasks sampled from some distribution throughout its lifetime (Abel *et al.*, 2018). We therefore consider a family of related tasks $\mathcal{M}$ with an absorbing set $\mathcal{G} \subseteq \mathcal{S}$ and restricted by the following assumption:

Assumption 3.3.1. *For all tasks in a set of tasks $\mathcal{M}$, (i) the tasks share the same state space, action space and transition dynamics, (ii) the transition dynamics are deterministic, and (iii) reward functions differ between tasks only on the absorbing set $\mathcal{G}$. That is, for all $M_1, M_2 \in \mathcal{M}$ with reward functions r_{M_1} and r_{M_2} respectively, we have that $r_{M_1}(s,a) = r_{M_2}(s,a) = r_{s,a} \in \mathbb{R}$ for all $s \in \mathcal{S} \setminus \mathcal{G}$ and $a \in \mathcal{A}$.*

Assumption 3.3.1 is similar to that of Todorov (2007) and identical to Van Niekerk *et al.* (2019), and imply that each task can be uniquely specified by its reward function. Note that because all tasks share the same transition dynamics, they must formally also share the same absorbing states. We consider the absorbing set as the set of all achievable goals in the environment, and each task is simply defined by how desirable each of those goals are. The assumption that the reward functions only differ on the absorbing set ensures the agent's experience before reaching goal states is consistent across all tasks.

Although we have placed restrictions on the reward functions, the above formulation still allows for a large number of tasks to be represented. Importantly, tasks with dense rewards in the absorbing set can be formulated under these restrictions.

3.4 Algebra of Tasks

In this section we formalise the logical composition of tasks.

3.4.1 Task Lattice

Having established how a lattice algebra abstracts the usual concept of disjunction and conjunction, we also use it to formalise the meaning of disjunction and conjunction of tasks. Since we have that tasks differ only in their reward functions, and the rewards over $\mathcal{S} \times \mathcal{A}$ are partially ordered with pointwise $\leq$ (the usual $\leq$ relation on $\mathbb{R}$), we have that the reward functions define a partial order on the tasks. The resulting partially ordered set of tasks is formally stated as follows:

Proposition 3.4.1. *Let $\mathcal{M}$ be a set of tasks, and $M_1, M_2 \in \mathcal{M}$ with reward functions r_{M_1}, r_{M_2} respectively. Then $(\mathcal{M}, \leq)$ is a partially ordered set with the relation $\leq$ given by*

$$M_1 \leq M_2 \textit{ if } r_{M_1}(s,a) \leq r_{M_2}(s,a) \textit{ for all } (s,a) \in \mathcal{S} \times \mathcal{A}.$$

Proof. Follows from the usual $\leq$ relation on $\mathbb{R}$. □

Since the rewards are real valued every pair has a least upper bound (their *sup*) and a greatest lower bound (their *inf*). Note how the resulting real functions after pointwise *inf* and *sup* are clearly still valid task rewards. Hence the poset of tasks $(\mathcal{M}, \leq)$ has a least upper bound $\sup\{M_1, M_2\} \in \mathcal{M}$ and a greatest lower bound $\inf\{M_1, M_2\} \in \mathcal{M}$ for any pair of task $M_1, M_2 \in \mathcal{M}$ (since they only differ on their rewards). The lattice $(\mathcal{M}, \vee, \wedge)$ induced by this partial order trivially follows with the binary operators $\vee$ and $\wedge$ given by $M_1 \vee M_2 := \sup\{M_1, M_2\}$ and $M_1 \wedge M_2 := \inf\{M_1, M_2\}$. We define these operators formally as follows:

Definition 3.4.1. *Let $\mathcal{M}$ be a set of tasks. The join and meet operators on $\mathcal{M}$ are respectively given by*

$$\begin{aligned}\vee :\ & \mathcal{M} \times \mathcal{M} \to \mathcal{M} \\ & (M_1, M_2) \mapsto (\mathcal{S}, \mathcal{A}, \rho, r_{M_1 \vee M_2}, \gamma), \textit{ where } r_{M_1 \vee M_2} : \mathcal{S} \times \mathcal{A} \to \mathbb{R} \\ & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (s,a) \mapsto \sup\{r_{M_1}(s,a), r_{M_2}(s,a)\},\end{aligned}$$

$$\begin{aligned}\wedge :\ & \mathcal{M} \times \mathcal{M} \to \mathcal{M} \\ & (M_1, M_2) \mapsto (\mathcal{S}, \mathcal{A}, \rho, r_{M_1 \wedge M_2}, \gamma), \textit{ where } r_{M_1 \wedge M_2} : \mathcal{S} \times \mathcal{A} \to \mathbb{R} \\ & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (s,a) \mapsto \inf\{r_{M_1}(s,a), r_{M_2}(s,a)\}.\end{aligned}$$

In fact $(\mathcal{M}, \vee, \wedge)$ forms a distributive lattice. Using the definitions of $\vee$ and $\wedge$, we state this as follows:

Proposition 3.4.2. *Let $\mathcal{M}$ be a set of tasks. Then $(\mathcal{M}, \vee, \wedge)$ is a distributive lattice.*

Proof. Follows from the properties of *inf* and *sup* and their distributivity. □

Given a non-empty finite set $\mathcal{O}$ of lower bounded subsets of tasks $\mathcal{N} \subset \mathcal{M}$, the task lattice $(\mathcal{M}, \vee, \wedge)$ gives us the principled way of specifying the disjunction of conjunctions,

$$\bigvee_{\mathcal{N} \in \mathcal{O}} \left(\bigwedge_{N \in \mathcal{N}} N \right) = (\mathcal{S}, \mathcal{A}, \rho, r_{\underset{\mathcal{O}}{\vee} \underset{\mathcal{N}}{\wedge}}, \gamma), \quad \text{where} \quad r_{\underset{\mathcal{O}}{\vee} \underset{\mathcal{N}}{\wedge}} : (s,a) \mapsto \sup_{\mathcal{N} \in \mathcal{O}} \left(\inf_{N \in \mathcal{N}} r_N(s,a) \right).$$

Similarly, given a non-empty finite set $\mathcal{O}$ of upper bounded subsets of tasks $\mathcal{N} \subset \mathcal{M}$, the conjunction of disjunctions is given by,

$$\bigwedge_{\mathcal{N} \in \mathcal{O}} \left(\bigvee_{N \in \mathcal{N}} N \right) = (\mathcal{S}, \mathcal{A}, \rho, r_{\underset{\mathcal{O}}{\wedge}\underset{\mathcal{N}}{\vee}}, \gamma), \quad \text{where} \quad r_{\underset{\mathcal{O}}{\wedge}\underset{\mathcal{N}}{\vee}} : (s,a) \mapsto \inf_{\mathcal{N} \in \mathcal{O}} \left(\sup_{N \in \mathcal{N}} r_N(s,a) \right).$$

Further more if $\mathcal{M}$ is bounded,[1] then it forms a complete distributive lattice. This means that the above holds true for any non-empty subset of $\mathcal{M}$.

Example 3.4.1. *Consider the simple grid world domain where an agent may be asked to go to any position on the xy-plane. The agent can move in any of the four cardinal directions at each timestep, but colliding with a wall leaves the agent in the same location. We add a 5th action for "stay" that the agent chooses to achieve goals. A goal position only becomes terminal if the agent chooses to stay in it. The transition dynamics are deterministic. The internal rewards (rewards for all non-terminal states) are r_{MIN} and the goal rewards (rewards on the absorbing set) range from r_{MIN} to r_{MAX}. That is, an agent receives a reward of r_{MIN} as it acts in the environment but when it chooses to stay at any location, it receives a reward between r_{MIN} and r_{MAX} depending on how close it is to the desired goal locations.*

Consider the specification of two tasks, M_L and M_D, in which an agent must navigate to the left and bottom of the xy-plane respectively. Figure 3.1 shows the rewards of the task specified by their disjunction and conjunction, which produces the task in which an agent must navigate to the left or bottom walls or to the corner of left and bottom walls respectively.

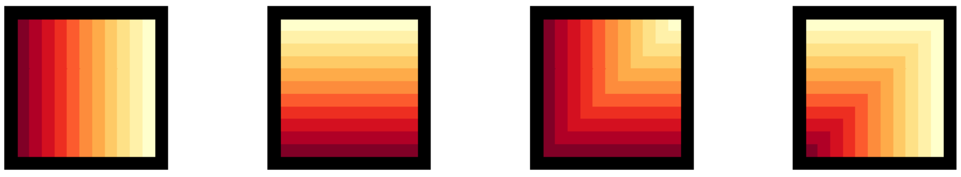

(a) M_L (b) M_D (c) $M_\text{L} \vee M_\text{D}$ (d) $M_\text{L} \wedge M_\text{D}$

Figure 3.1: Rewards at terminal states for the disjunction and conjunction of tasks in gridworld domain (any position becomes terminal if the agent chooses to stay in it). All internal rewards are r_MIN. Figures (a)-(b) shows the dense goal rewards for the left and down tasks. (c) shows the rewards of their disjunction. (d) shows the rewards of their conjunction.

3.4.2 De Morgan Task Algebra

Having formalised the meaning of disjunction and conjunction of tasks, the next natural question is what is the meaning of the negation of a task? A De Morgan algebra enables us to define it by adding minimal required properties that encapsulates the desired behaviour of a negation. We first make the following assumption:

Assumption 3.4.1. *For all tasks in a set of tasks $\mathcal{M}$, their reward function is bounded by $[r_{MIN}, r_{MAX}] \subset \mathbb{R}$*

[1]For simplicity, we adopt the convention of Grätzer (2011) of referring to an algebraic structure (e.g the lattice $(\mathcal{M}, \vee, \wedge)$) by the set on which it is defined ($\mathcal{M}$), when that doesn't lead to any ambiguity.

Any set of tasks $\mathcal{M}$ satisfying Assumption 3.4.1 has bounds $\mathcal{M}_{SUP}, \mathcal{M}_{INF} \in \mathcal{M}$ whose respective reward functions are

$$\begin{aligned} r_{\mathcal{M}_{SUP}} : \ & \mathcal{S} \times \mathcal{A} \to \mathbb{R} \\ & (s,a) \mapsto \sup_{M \in \mathcal{M}} r_M(s,a), \end{aligned} \qquad \begin{aligned} r_{\mathcal{M}_{INF}} : \ & \mathcal{S} \times \mathcal{A} \to \mathbb{R} \\ & (s,a) \mapsto \inf_{M \in \mathcal{M}} r_M(s,a). \end{aligned}$$

We now define the negation of a task as follows:

Definition 3.4.2. *Let $\mathcal{M}$ be a bounded set of tasks with bounds $\mathcal{M}_{SUP}, \mathcal{M}_{INF} \in \mathcal{M}$. Define the negation operator as*

$$\begin{aligned} \neg : \ & \mathcal{M} \to \mathcal{M} \\ & M \mapsto (\mathcal{S}, \mathcal{A}, \rho, r_{\neg M}, \gamma), \ \textit{where } r_{\neg M} : \ \mathcal{S} \times \mathcal{A} \to \mathbb{R} \\ & \qquad\qquad (s,a) \mapsto (r_{\mathcal{M}_{SUP}}(s,a) + r_{\mathcal{M}_{INF}}(s,a)) - r_M(s,a). \end{aligned}$$

The intuition for this definition is straightforward. For example if for a given task taking an action at a given state gives the smallest reward, then the opposite task should give the highest reward for that state-action. Note that for $M \in \mathcal{M}$, we have $(\mathcal{S}, \mathcal{A}, \rho, r_{\neg M}, \gamma) \in \mathcal{M}$ since $r_{\neg M}$ is a valid task reward function.

We finally formalize the interaction of the negation of tasks with the conjunction and disjunction of tasks as follows:

Proposition 3.4.3. *Let $\mathcal{M}$ be a set of tasks satisfying Assumption 3.4.1. Then $(\mathcal{M}, \vee, \wedge, \neg, \mathcal{M}_{SUP}, \mathcal{M}_{INF})$ is a De Morgan algebra.*

Proof. Let $M_1, M_2 \in \mathcal{M}$. We show that $\neg, \vee, \wedge$ satisfy the De Morgan algebra properties (i) – (vii).

(i)–(v): These follow from the properties of *inf* and *sup*.

(vi): This follows from the bounds $\mathcal{M}_{SUP}, \mathcal{M}_{INF} \in \mathcal{M}$ which are guaranteed to exist due to Assumption 3.4.1.

(vii): The first condition easily follows from the definition of $\neg$. For the second condition, let $r_{\neg(M_1 \vee M_2)}$ be the reward function for $\neg(M_1 \vee M_2)$. Then for all (s,a) in $\mathcal{S} \times \mathcal{A}$,

$$\begin{aligned} r_{\neg(M_1 \vee M_2)}(s,a) &= (r_{\mathcal{M}_{SUP}}(s,a) + r_{\mathcal{M}_{INF}}(s,a)) - \sup_{M \in \{M_1, M_2\}} r_M(s,a) \\ &= (r_{\mathcal{M}_{SUP}}(s,a) + r_{\mathcal{M}_{INF}}(s,a)) + \inf_{M \in \{M_1, M_2\}} -r_M(s,a) \\ &= \inf_{M \in \{M_1, M_2\}} (r_{\mathcal{M}_{SUP}}(s,a) + r_{\mathcal{M}_{INF}}(s,a)) - r_M(s,a) \\ &= r_{\neg M_1 \wedge \neg M_2}(s,a). \end{aligned}$$

Thus $\neg(M_1 \vee M_2) = \neg M_1 \wedge \neg M_2$.

□

We can now specify arbitrary disjunction, conjunction, and negation of tasks.

Example 3.4.2. *Consider the simple grid world domain introduced in Example 3.4.1 where an agent may be asked to go to any position on the xy-plane. Further consider the specification of two tasks, M_L and M_D, in which an agent must navigate to the left and bottom of the xy-plane respectively. Figure 3.2 shows the rewards of the tasks specified by their negation, $\neg M_L = M_R$ and $\neg M_D = M_T$, in which an agent must navigate to the right and top of the xy-plane respectively. This shows that the negation defined above does indeed have the expected semantics. The figure also shows that arbitrary disjunction, conjunction, and negation of tasks also produces tasks with the desired semantics.*

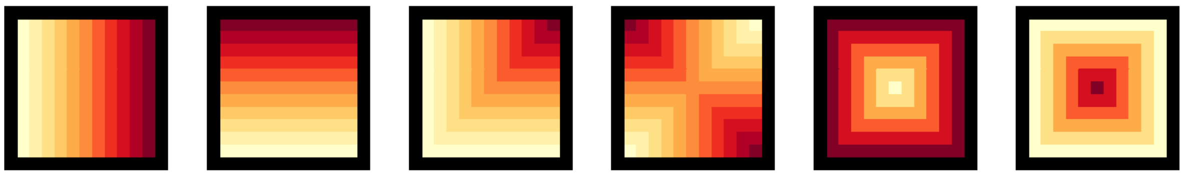

(a) $\neg M_{\text{L}}$ (b) $\neg M_{\text{D}}$ (c) $M_{\text{L}} \overline{\vee} M_{\text{D}}$ (d) $M_{\text{L}} \veebar M_{\text{D}}$ (e) $\begin{matrix}(M_{\text{L}} \vee \neg M_{\text{L}}) \\ \vee \\ (M_{\text{D}} \vee \neg M_{\text{D}})\end{matrix}$ (f) $\begin{matrix}(M_{\text{L}} \wedge \neg M_{\text{L}}) \\ \wedge \\ (M_{\text{D}} \wedge \neg M_{\text{D}})\end{matrix}$

Figure 3.2: Rewards at terminal states for the composition of tasks in gridworld domain (any position becomes terminal if the agent chooses to stay in it). All internal rewards are r_{MIN}. Figures (a)-(b) shows the rewards for the right and top tasks, specified by the negation of the left and down tasks respectively. (c) shows the rewards of the task specified by the negation of the disjunction of the left and down tasks—$M_{\text{L}} \overline{\vee} M_{\text{D}} := \neg(M_{\text{L}} \vee M_{\text{D}})$. (d) shows the rewards of the task specified by the exclusive disjunction of the left and down tasks—$M_{\text{L}} \veebar M_{\text{D}} := (M_{\text{L}} \vee M_{\text{D}}) \wedge \neg(M_{\text{L}} \wedge M_{\text{D}})$. (e) shows the rewards of the task specified by the disjunction of the left, right, down and top tasks. (f) shows the rewards of the task specified by the conjunction of the left, right, down and top tasks.

3.4.3 Boolean Task Algebra

While the De Morgan task algebra allows us to do arbitrary logical compositions of tasks with dense goal rewards, it gives no guarantees on some desired properties of logics. In particular these task compositions do not always satisfy the laws of the excluded middle, $M_1 \vee \neg M_1 = \mathcal{M}_{SUP}$, and of non contradiction, $M_1 \wedge \neg M_1 = \mathcal{M}_{INF}$. This can be clearly seen in say Figure 3.2(f), in which the agent needs to go *to the left wall and not to the left wall* and *to the bottom wall and not to the bottom wall.* While in this case because of the choice of dense rewards it produces a meaningful task — the task in which the agent needs to go to the center — in general we may want to guarantee that contradicting task specifications are equally meaningless. In this section we show that by restricting the set of tasks to those with sparse rewards, we obtain the full properties of logics on them.

The following sparseness assumption ensures tasks have a Boolean nature. Assumption 3.4.2 says that tasks are defined by goals which are either desirable or not.

Assumption 3.4.2. *For all tasks in a set of tasks $\mathcal{M}$ which adhere to Assumption 3.4.1 , the set of possible terminal rewards consists of only two values. That is, for all (g, a) in $\mathcal{G} \times \mathcal{A}$, we have that $r(g, a) \in \{r_\varnothing, r_\mathcal{U}\} \subset \mathbb{R}$ with $r_\varnothing \leq r_\mathcal{U}$.*[2]

We can now formalize logics on the set of tasks.

[2]Since Assumption 3.4.2 includes Assumption 3.4.1, any set of tasks satisfying it forms a De Mogran task algebra according to Proposition 3.4.3.

Proposition 3.4.4. *Let $\mathcal{M}$ be a set of tasks satisfying Assumption 3.4.2. Then $(\mathcal{M}, \vee, \wedge, \neg, \mathcal{M}_{SUP}, \mathcal{M}_{INF})$ is a Boolean algebra.*

Proof. Let $M_1, M_2 \in \mathcal{M}$. We show that $\neg, \vee, \wedge$ satisfy the Boolean properties (i) – (vii).

(i)–(vi): These follow from the De Morgan task algebra since $\mathcal{M}$ satisfies its assumptions.

(vii): Let $r_{M_1 \wedge \neg M_1}$ be the reward function for $M_1 \wedge \neg M_1$. Then for all (s, a) in $\mathcal{S} \times \mathcal{A}$,

$$\begin{aligned} r_{M_1 \wedge \neg M_1}(s,a) &= \begin{cases} \inf\{r_{M_1}(s,a), (r_{\mathcal{U}} + r_{\varnothing}) - r_{M_1}(s,a)\}, & \text{if } s \in \mathcal{G} \\ \inf\{r_{s,a}, (r_{s,a} + r_{s,a}) - r_{s,a}\}, & \text{otherwise.} \end{cases} \\ &= \begin{cases} r_{\varnothing}, & \text{if } s \in \mathcal{G} \text{ and } r_{M_1}(s,a) = r_{\mathcal{U}} \\ r_{\varnothing}, & \text{if } s \in \mathcal{G} \text{ and } r_{M_1}(s,a) = r_{\varnothing} \\ r_{s,a}, & \text{otherwise.} \end{cases} \\ &= r_{\mathcal{M}_{INF}}(s,a). \end{aligned}$$

Thus $M_1 \wedge \neg M_1 = \mathcal{M}_{INF}$, and similarly $M_1 \vee \neg M_1 = \mathcal{M}_{SUP}$.

□

Example 3.4.3. *Consider the simple grid world domain introduced in Example 3.4.1 where an agent may be asked to go to any position on the xy-plane. We modify it to satisfy Assumption 3.4.2 by restricting the goal rewards (rewards on the absorbing set) to r_{MIN} or r_{MAX}. Consider the specification of two tasks, M_L and M_D, in which an agent must navigate to the left and bottom of the xy-plane respectively. Figure 3.3 shows the rewards of the tasks specified by their logical composition. Note how composing tasks specified using sparse rewards is semantically different than when tasks are specified using dense rewards. For example the negation of the left task here now means "do not go to the left wall" (instead of "go to the right wall" as seen in the dense case). Also meaningless compositions like the conjunction of "go to the left wall" and "do not go to the left wall" now give the expected lower bound task, which says no goal is desirable.*

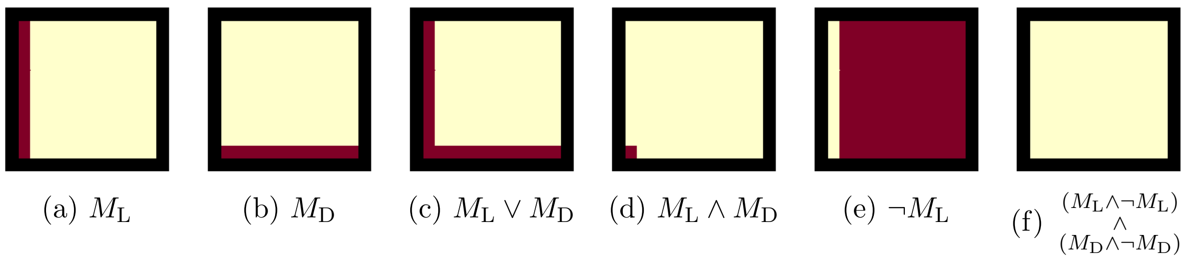

(a) M_L (b) M_D (c) $M_\text{L} \vee M_\text{D}$ (d) $M_\text{L} \wedge M_\text{D}$ (e) $\neg M_\text{L}$ (f) $\frac{(M_\text{L} \wedge \neg M_\text{L})}{\wedge \, (M_\text{D} \wedge \neg M_\text{D})}$

Figure 3.3: Showing composition of boolean tasks in gridworld domain. Figures (a)-(b) shows the Boolean goal rewards for the left and down tasks, (c)-(f) shows the rewards of the composed tasks.

Finally we summarize the main differences between the developed task algebras in Table 3.1. In the next section we show that an additional benefit of the Boolean task algebra is that it gives us the notion of base tasks.

	Unbounded task space	Dense goal rewards	Negation of tasks	Full logics on tasks
Task lattice	✓	✓		
De Morgan task algebra		✓	✓	
Boolean task algebra			✓	✓

Table 3.1: Difference between the benefits obtained from each algebraic structure.

3.5 Between Task and Power Set Boolean Algebras

Having established a Boolean algebra over tasks, we show that there exists an equivalence between them and a power set algebra. We note that Assumption 3.4.2 establishes a bijection F between the set of tasks $\mathcal{M}$ and the power-set $\mathcal{P}(\mathcal{G} \times \mathcal{A})$, given by

$$\begin{aligned} F:\ & \mathcal{P}(\mathcal{G} \times \mathcal{A}) \to \mathcal{M} \\ & H \mapsto (\mathcal{S}, \mathcal{A}, \rho, r_H, \gamma), \text{ where } r_H:\ \mathcal{S} \times \mathcal{A} \to \mathbb{R} \\ & (s,a) \mapsto \begin{cases} r_{\mathcal{U}}, & \text{if } (s,a) \in H \\ r_{\varnothing}, & \text{if } (s,a) \notin H \text{ and } s \in \mathcal{G} \\ r_{s,a}, & \text{otherwise.} \end{cases} \end{aligned}$$

The Boolean task algebra together with the bijection between tasks $\mathcal{M}$ and the power-set $\mathcal{P}(\mathcal{G} \times \mathcal{A})$ gives us the following result.

Proposition 3.5.1. *Let $\mathcal{M}$ be a set of tasks satisfying Assumption 3.4.2. Then the Boolean task algebra on $\mathcal{M}$ is isomorphic to the power set Boolean algebra on $\mathcal{P}(\mathcal{G} \times \mathcal{A})$.*

Proof. This follows from the bijection F and the fact that it is clearly homomorphic. □

This means that all the results that hold for power set Boolean algebras now holds for Boolean task algebras. In particular consider the case of a set of tasks $\mathcal{M}$ with goal rewards that are sparse and independent of the goal actions. That is, for all tasks in $\mathcal{M}$ goals are either desirable or not. Then the respective Boolean task sub-algebra is isomorphic to the power set algebra on the absorbing set $\mathcal{G}$ with the isomorphism F given by,

$$\begin{aligned} F:\ & \mathcal{P}(\mathcal{G}) \to \mathcal{M} \\ & H \mapsto (\mathcal{S}, \mathcal{A}, \rho, r_H, \gamma), \text{ where } r_H:\ \mathcal{S} \times \mathcal{A} \to \mathbb{R} \\ & (s,a) \mapsto \begin{cases} r_{\mathcal{U}}, & \text{if } s \in H \\ r_{\varnothing}, & \text{if } s \notin H \text{ and } s \in \mathcal{G} \\ r_{s,a}, & \text{otherwise.} \end{cases} \end{aligned}$$

This means that for $\mathcal{M}$ with finite $\mathcal{G}$, we need only a logarithmic number of base tasks (generators) $\lceil \log_2 |\mathcal{G}| \rceil$ (for $|\mathcal{G}| > 1$) to specify an exponential number of composed tasks $2^{|\mathcal{G}|} = |\mathcal{M}|$.

Example 3.5.1. *Consider a Four Rooms domain (Sutton* et al.*, 1999), where an agent must navigate in a grid world to particular rooms. Similarly to the gridworld in Example 3.4.1, the agent can move in any of the four cardinal directions at each timestep, but colliding with a wall leaves the agent in the same location. There is a 5th action for "stay"* *that the agent chooses to achieve goals. A goal position (center of a room) only becomes terminal if the agent chooses to stay in it. The transition dynamics are deterministic. The internal rewards (rewards for all non-terminal states) are r_{MIN} and the goal rewards (rewards on the absorbing set) are sparse (r_{MIN} or r_{MAX}). Figure 3.4 illustrates the layout of the environment and the goals the agent must reach.*

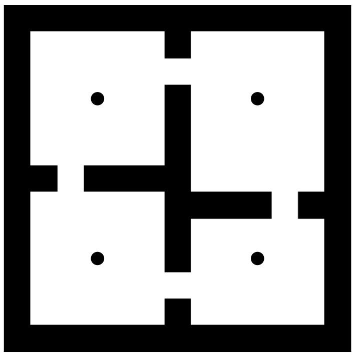


Figure 3.4: The layout of the Four Rooms domain. The circles indicate goals the agent must reach. We will refer to the goals as `top-left`, `top-right`, `bottom-left`, and `bottom-right`.

We can select a minimal set of base tasks (generator tasks) by assigning each goal a binary number, and then using the columns of the table to select the tasks. Since the set of achievable goals (the absorbing set) is finite, we can do this assignment using a Boolean table. We first assign labels to the individual goals by defining a well order over the set $\mathcal{G}$. The number of base tasks induced by this well order is $\lceil |\log_2 \mathcal{G}| \rceil = 2$ since $|\mathcal{G}| = 4$. Table 3.2 illustrates how different well orders on $\mathcal{G}$ leads to different choices of base tasks.

Goals	M_D	M_R
top-left	0	0
top-right	0	1
bottom-left	1	0
bottom-right	1	1

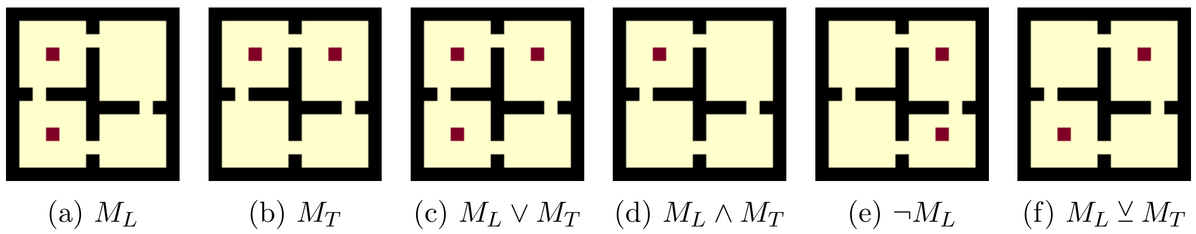
(a) Goals labled by the well order $\leq$ given by: $bottom-right \leq bottom - left \leq top-right \leq top-left$

Goals	M_T	M_L
bottom-right	0	0
bottom-left	0	1
top-right	1	0
top-left	1	1

(b) Goals labled by the well order $\leq$ given by: $top-left \leq top-right \leq bottom-left \leq bottom-right$

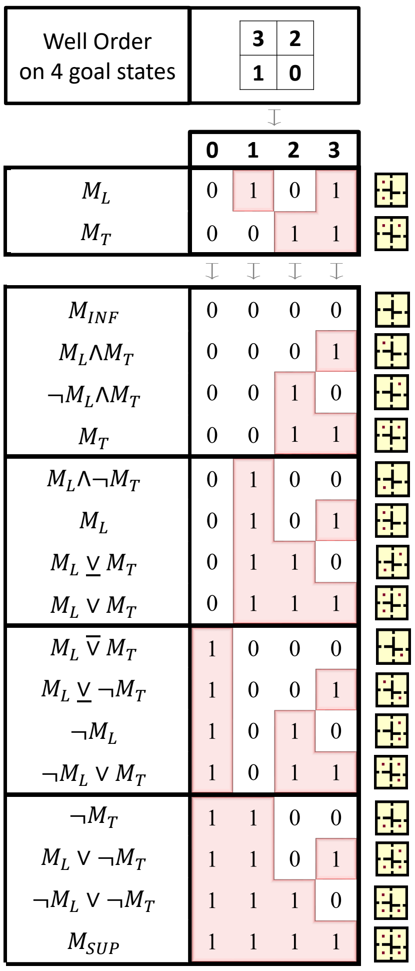
Table 3.2: Base tasks induced by various well orders on $\mathcal{G}$. Each column represents a base task, where **0** or **1** for goal g on task M means respectively reward of $r_M(g,a) = r_{\text{MIN}}$ or $r_M(g,a) = r_{\text{MAX}}$ $\forall a \in \mathcal{A}$. Note that the goal rewards are enough to specify the tasks since the internal rewards are the same across all tasks.

Consider for example the well order on $\mathcal{G}$ shown in Table 3.2(b). The base tasks induced are M_L and M_T, in which an agent must navigate to the 2 left rooms (rooms 1 and 3) and the 2 top rooms (rooms 2 and 3) respectively. Figure 3.5 shows the rewards of the tasks specified by some of their logical compositions. Figure 3.6 shows the Boolean table and Hasse diagram for all the $2^{2^k} = 16$ tasks generated by the $k = 2$ base tasks, which spans the whole set of tasks, $|\mathcal{M}| = 2^{|\mathcal{G}|} = 16$.



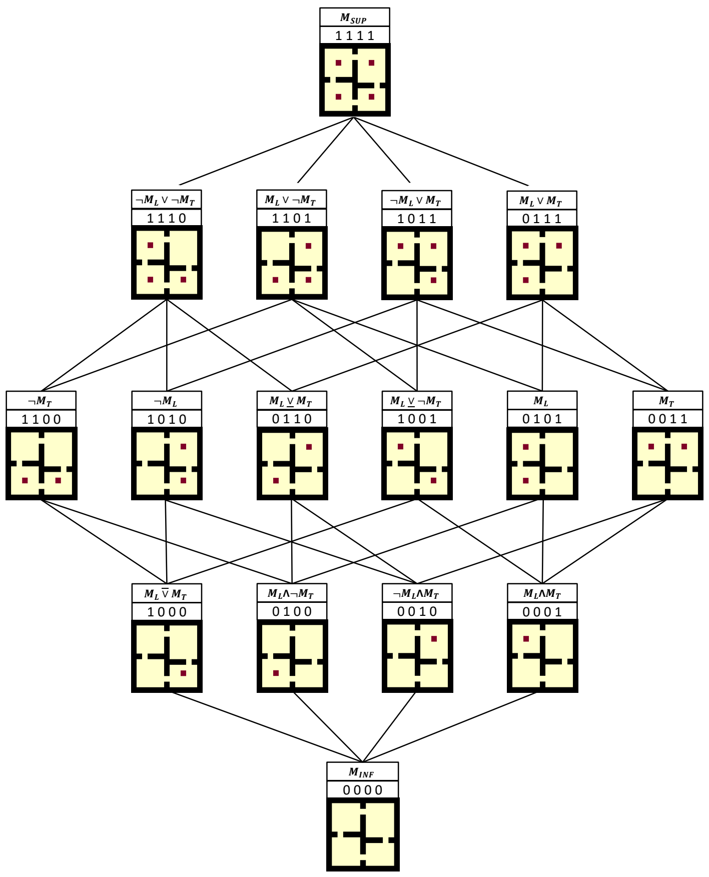
(a) M_L (b) M_T (c) $M_L \vee M_T$ (d) $M_L \wedge M_T$ (e) $\neg M_L$ (f) $M_L \veebar M_T$

Figure 3.5: Showing task composition of base tasks in four-rooms domain. Figures (a)-(b) shows the rewards for the base tasks, (c)-(f) shows the rewards of the composed tasks. $\veebar$ represents exclusive disjunction.



Well Order on 4 goal states	3 2 / 1 0			
	0	**1**	**2**	**3**
M_L	0	1	0	1
M_T	0	0	1	1
M_{INF}	0	0	0	0
$M_L \wedge M_T$	0	0	0	1
$\neg M_L \wedge M_T$	0	0	1	0
M_T	0	0	1	1
$M_L \wedge \neg M_T$	0	1	0	0
M_L	0	1	0	1
$M_L \veebar M_T$	0	1	1	0
$M_L \vee M_T$	0	1	1	1
$M_L \barwedge M_T$	1	0	0	0
$M_L \veebar \neg M_T$	1	0	0	1
$\neg M_L$	1	0	1	0
$\neg M_L \vee M_T$	1	0	1	1
$\neg M_T$	1	1	0	0
$M_L \vee \neg M_T$	1	1	0	1
$\neg M_L \vee \neg M_T$	1	1	1	0
M_{SUP}	1	1	1	1

(a) Boolean table of base and composed tasks.



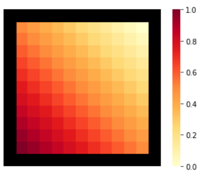
(b) Hasse diagram of the Boolean task algebra.

Figure 3.6: Boolean table and Hasse diagram for the four-rooms domain. (a) shows a well order on $\mathcal{G}$ that labels the goals, the induced base tasks from labeling the goals, the logical expressions for all 16 compositions of the base tasks, their Boolean values and rewards. (b) illustrates the Boolean task algebra showing the rewards for all 16 tasks in $\mathcal{M}$, together with the Boolean values and logical expressions that generates them from the base tasks.

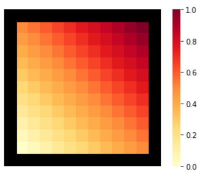
3.6 Problem with Standard Value Functions for Zero-Shot Composition

Having established the logical composition of tasks in the previous sections, we note that standard value functions are insufficient to solve them in a zero-shot manner. In particular they are insufficient to do zero-shot conjunction of tasks. To understand why, consider two tasks that have multiple different goals but at least one common goal. Clearly, there is a meaningful conjunction between them—namely, achieving the common goal. Now consider an agent that learns standard value functions for both tasks, and which is then required to solve their conjunction without further learning. Note that this is impossible in general, since the standard value function for each task only represents the value of each state with respect to the *nearest* goal. That is, for all states where the nearest goal for each task is *not* the common goal, the agent has no information about that common goal.

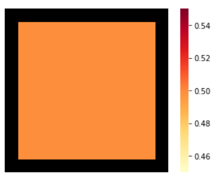
While previous work has used the average reward function to approximate the conjunction operator (Haarnoja *et al.*, 2018; Hunt *et al.*, 2019; Van Niekerk *et al.*, 2019), we note that the tasks specified by averaging the rewards quickly diverges from the task specified by the *min* of rewards (the conjunction). In particular Figure 3.7 shows how the average reward task becomes very different to the *min* rewards task after only 3 operations. This highlights the importance of working towards zero-shot composition using the true logical operators, as it enable multiple arbitrary logical compositions while retaining the meaning of the specified tasks.



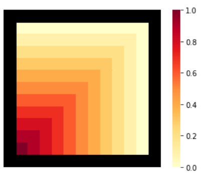
(a) $0.5(r_{M_{\mathrm{L}}} + r_{M_{\mathrm{D}}})$



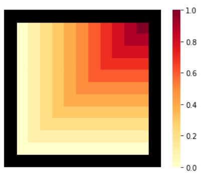
(b) $0.5(r_{M_{\mathrm{T}}} + r_{M_{\mathrm{R}}})$



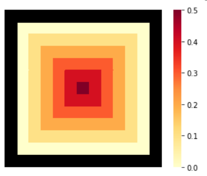
(c) $0.5\begin{pmatrix} 0.5(r_{M_{\mathrm{L}}}+r_{M_{\mathrm{D}}}) \\ + \\ 0.5(r_{M_{\mathrm{T}}}+r_{M_{\mathrm{R}}}) \end{pmatrix}$



(d) $\min\{r_{M_{\mathrm{L}}}, r_{M_{\mathrm{D}}}\}$



(e) $\min\{r_{M_{\mathrm{T}}}, r_{M_{\mathrm{R}}}\}$



(f) $\min\left\{\begin{matrix} \min\{r_{M_{\mathrm{L}}},r_{M_{\mathrm{D}}}\}, \\ \min\{r_{M_{\mathrm{T}}},r_{M_{\mathrm{R}}}\} \end{matrix}\right\}$

Figure 3.7: Consider 4 tasks, M_{L}, M_{R}, M_{D}, and M_{T} , in which an agent must navigate to the left, right, down, and top regions of an xy-plane respectively. The top row shows conjunctions approximated using average of rewards while the bottom one shows true conjunctions using the *min* of rewards. From left to right we plot the reward for entering a region of the goal space for the conjunction of M_{L} and M_{D}, conjunction of M_{T} and M_{R}, and conjunction of (M_{L} and M_{D}) and (M_{T} and M_{R}). Note that by averaging reward, terminal states quickly become equally rewarded, losing the meaning of the true conjunction.

3.7 Conclusion

In this chapter we have formally established the logical composition of tasks. This gives us a structured and simple way of obtaining the MDP that models a task specified as the logical composition of other tasks. In the case of tasks with sparse rewards, we introduce a notion of base tasks and demonstrate how they can be obtained from Boolean tables. These base tasks are proven sufficient to specify any other task in an environment. Finally we showed that the knowledge represented by standard value functions is in fact too task specific, and hence does not encode sufficient information to be able to solve new tasks zero-shot. In the next chapter we introduce a new type of value function that is much richer in its knowledge representation.

Chapter 4

Extended Value Functions

4.1 Introduction

In the previous chapter we established how to formally compose tasks but showed that standard value functions are insufficient to solve them zero-shot. The argument here was that standard value functions only learn how to achieve the nearest goal and hence do not encode information about other goals which may be valuable for other tasks. We address this issue here by introducing a value function that learns about various achievable goals in the environment. This encodes the intuition that an agent should learn as much as possible from its experience. It should learn not just how to achieve the best goal, but also what behaviours achieve the lesser goals or even the bad ones.

The main contributions of this chapter are as follows:

- We introduce a new type of goal oriented value function called extended value function (EVF) that learns a diversity of solutions for a single task specification.
- We prove that the standard value function for a task can be recovered from this goal oriented value function by simply maximizing goals.
- We prove that for sample classes of MDPs that naturally model goal-reaching tasks, the introduced goal oriented value function in fact learns how to achieve any terminal state in the environment, irrespective of whether it is desired or not for the given task. That is, for a given task it learns about both its good (positively rewarded) and bad (negatively rewarded) terminal states.
- We introduce a new learning method called extended q-learning to learn this new goal oriented value function without the reward function of tasks being provided. We similarly modify Deep-Q-Networks to learn these new value functions in the function-approximation case.

This chapter is structured as follows. In Section 4.2 we develop the extended value functions and prove some of its key properties. Since these are goal-oriented value functions, we discuss some closely related work on other types of goal oriented value functions. In Section 4.3 we propose methods for learning the extended value functions. Finally in Section 4.4 we run relevant experiments for the tabular and function-approximation cases.

4.2 Theory for Extended Value Functions

As shown in Section 3.6, the reward and value functions described in Section 2.2 are insufficient to solve tasks specified by the lattice algebras developed in Section 3.4. We therefore extend these to define a goal-oriented reward and value function that captures the criterion that the agent needs to learn not just about the best goal, but also about all other goals.

Let $M = (\mathcal{S}, \mathcal{A}, p, r, \gamma)$ be an MDP with an absorbing set $\mathcal{G} \subseteq \mathcal{S}$ and whose rewards are bounded by $[r_{\text{MIN}}, r_{\text{MAX}}]$. We define its extended reward function as follows:

Definition 4.2.1. *The extended reward function $\bar{r} : \mathcal{S} \times \mathcal{G} \times \mathcal{A} \to \mathbb{R}$ is given by the mapping*

$$(s, g, a) \mapsto \begin{cases} \bar{r}_{MIN} & \text{if } g \neq s \in \mathcal{G} \\ r(s, a) & \text{otherwise,} \end{cases} \tag{4.1}$$

where $\bar{r}_{MIN} \leq r_{MIN}$.

Because we require that tasks share the same transition dynamics, we also require that the absorbing set of states is shared. Thus the extended reward function adds the extra constraint that, if the agent enters a terminal state for a *different* task, it should receive the smallest reward possible. The goal of the agent now is to compute an extended Markov policy $\bar{\pi} : \mathcal{S} \times \mathcal{G} \to Pr(\mathcal{A})$ that optimally solves any task relevant goal. A given extended policy $\bar{\pi}$ is characterised by an extended state-value function defined as follows:

Definition 4.2.2. *The extended state-value function $\bar{V}^{\bar{\pi}} : \mathcal{S} \times \mathcal{G} \to \mathbb{R}$ is given by the mapping*

$$(s, g) \mapsto \mathbb{E}_{\bar{\pi}} \left[\gamma \sum_{t=0}^{\infty} \bar{r}(s_t, g, a_t) \right]. \tag{4.2}$$

This specifies the expected return obtained under π starting from state s. Similarly we define the extended Q-value function, $\bar{Q}^{\bar{\pi}}(s, g, a)$, which defines the expected return obtained by executing a from s, and thereafter following $\bar{\pi}$ to reach g. We define this as follows:

Definition 4.2.3. *The extended action-value function $\bar{Q}^{\bar{\pi}} : \mathcal{S} \times \mathcal{G} \times \mathcal{A} \to \mathbb{R}$ is given by the mapping*

$$(s, g, a) \mapsto \mathbb{E}_{\bar{\pi}} \left[\bar{r}(s, g, a) + \gamma \sum_{t=1}^{\infty} \bar{r}(s_t, g, a_t) \right]. \tag{4.3}$$

Since for each g these extended value functions are equivalent to regular value functions, it follows that they satisfy the following Bellman equations,

$$\bar{V}^{\bar{\pi}}(s, g) = \sum_{a \in \mathcal{A}} \bar{\pi}(a, s, g) \left(\bar{r}(s, g, a) + \gamma \sum_{s' \in \mathcal{S}} p(s, a, s') \bar{V}^{\bar{\pi}}(s', g) \right), \tag{4.4}$$

$$\bar{Q}^{\bar{\pi}}(s, g, a) = \bar{r}(s, g, a) + \gamma \sum_{s' \in \mathcal{S}} p(s, a, s') \sum_{a \in \mathcal{A}} \bar{\pi}(a, s, g) \bar{Q}^{\bar{\pi}}(s', g, a). \tag{4.5}$$

In fact all standard results on regular value functions also hold here by extension. This can be shown by simply noting that each $g \in \mathcal{G}$ corresponds to a well defined MDP $M_g := (\mathcal{S}, \mathcal{A}, \rho, r_{M_g}, \gamma)$ with reward function $r_{M_g}(s, a) := \bar{r}_M(s, g, a)$, hence that the corresponding set of regular value functions defines the extended value function. In particular we have that for all goal-oriented MDPs there exist an optimal deterministic policy $\bar{\pi}^* \geq \bar{\pi}, \forall \bar{\pi}$, and unique optimal value functions $\bar{V}^*$ and $\bar{Q}^*$, such that

$$\bar{V}^{\bar{\pi}^*}(s,g) = \bar{V}^*(s,g) = \max_{\bar{\pi}} \bar{V}^{\bar{\pi}}(s,g) \;\; \forall \bar{\pi}^*,$$

$$\bar{Q}^{\bar{\pi}^*}(s,g,a) = \bar{Q}^*(s,g,a) = \max_{\bar{\pi}} \bar{Q}^{\bar{\pi}}(s,g) \;\; \forall \bar{\pi}^*.$$

These give rise to the following Bellman optimality equations for extended value functions,

$$\bar{V}^*(s,g) = \max_{a \in \mathcal{A}} \left(\bar{r}(s,g,a) + \gamma \sum_{s' \in \mathcal{S}} p(s,a,s') \bar{V}^*(s',g) \right), \tag{4.6}$$

$$\bar{Q}^*(s,g,a) = \bar{r}(s,g,a) + \gamma \sum_{s' \in \mathcal{S}} p(s,a,s') \max_{a \in \mathcal{A}} \bar{Q}^*(s',g,a). \tag{4.7}$$

The corresponding Bellman operator and Bellman optimality operator on extended value functions are respectively given by,

$$\left[\mathcal{T}^{\bar{\pi}} \bar{V}^{\bar{\pi}}\right](s,g) = \bar{r}(s,g,\bar{\pi}(s,g)) + \gamma \sum_{s' \in \mathcal{S}} p(s,\bar{\pi}(s,g),s') \bar{V}^{\bar{\pi}}(s',g), \tag{4.8}$$

$$\left[\mathcal{T} \bar{V}\right](s,g) = \max_{a \in \mathcal{A}} \left[\bar{r}(s,g,a) + \gamma \sum_{s' \in \mathcal{S}} p(s,a,s') \bar{V}(s',g) \right]. \tag{4.9}$$

Similarly to regular value functions, we note that $\bar{\pi}^*$ and $\bar{V}^*$ can be obtained by acting greedily on $\bar{Q}^*$. That is,

$$\bar{\pi}^*(a,s,g) = \begin{cases} 1 & a = \arg\max_{a' \in \mathcal{A}} \bar{Q}^*(s,g,a') \\ 0 & \text{otherwise,} \end{cases}$$

$$\bar{V}^*(s,g) = \max_{a \in \mathcal{A}} \bar{Q}^*(s,g,a).$$

A main property of EVFs is that the standard reward functions and value functions can be recovered from their extended versions through the following theorem.

Theorem 4.2.1. *Let $r_M, \bar{r}_M, Q^*_M, \bar{Q}^*_M$ be the reward function, extended reward function, optimal Q-value function, and optimal extended Q-value function for a task M in $\mathcal{M}$. Then for all (s,a) in $\mathcal{S} \times \mathcal{A}$, we have*
(i) $r_M(s,a) = \max_{g \in \mathcal{G}} \bar{r}_M(s,g,a)$, and (ii) $Q^*_M(s,a) = \max_{g \in \mathcal{G}} \bar{Q}^*_M(s,g,a)$.

Proof.

(i):

$$\begin{aligned} \max_{g \in \mathcal{G}} \bar{r}_M(s,g,a) &= \begin{cases} \max\{\bar{r}_{\text{MIN}}, r_M(s,a)\}, & \text{if } s \in \mathcal{G} \\ \max_{g \in \mathcal{G}} r_M(s,a), & \text{otherwise.} \end{cases} \\ &= r_M(s,a) \qquad (\bar{r}_{\text{MIN}} \leq r_{\text{MIN}} \leq r_M(s,a) \text{ by definition}). \end{aligned}$$

(ii): Each g in $\mathcal{G}$ can be thought of as defining an MDP $M_g := (\mathcal{S}, \mathcal{A}, \rho, r_{M_g})$ with reward function $r_{M_g}(s,a) := \bar{r}_M(s,g,a)$ and optimal Q-value function $Q^*_{M_g}(s,a) = \bar{Q}^*_M(s,g,a)$. Then using (i) we have $r_M(s,a) = \max\limits_{g \in \mathcal{G}} r_{M_g}(s,a)$ and from Van Niekerk *et al.* (2019, Corollary 1), we have that $Q^*_M(s,a) = \max\limits_{g \in \mathcal{G}} Q^*_{M_g}(s,a) = \max\limits_{g \in \mathcal{G}} \bar{Q}^*_M(s,g,a)$.

□

In the same way, we can also recover the optimal policy from the extended action-value functions by first applying Theorem 4.2.1, and acting greedily with respect to the resulting action-value function.

For the rest of this work we focus on two classes of MDPs that naturally model goal reaching tasks, namely deterministic shortest path problems (Bertsekas & Tsitsiklis, 1991) and discounted goal reaching problems.

4.2.1 Related Work

EVFs are similar to DG functions (Kaelbling, 1993), except here we use task-dependent reward functions, as opposed to measuring distance between states. It is also similar to universal value function approximators (UVFAs) (Schaul *et al.*, 2015), but differs in that it uses the extended reward function definition. UVFAs use general value functions (Sutton *et al.*, 2011) with goal based reward functions, where each goal specify a separate task, and to act in the environment the goal for the desired task need to be provided. In contrast we use the standard reward function of a single task. The extended reward function we define is implicit to the agent and the agent acts in the environment by using Theorem 4.2.1.

Perhaps the closest related work is hindsight experience replay (Andrychowicz *et al.*, 2017), where an agent learns to achieve multiple goals for a single overall task. However they still use UVFAs with goal based reward functions that are provided in advance. This means the agent still needs to be given the goal to be achieved since that defines the task to be solved.

4.2.2 EVFs for Deterministic Shortest Path Tasks

Deterministic shortest path problems represent the class of deterministic undiscounted MDPs with an absorbing set $\mathcal{G} \subseteq \mathcal{S}$. Since it considers undiscounted MDPs, the value function of policies that reach $\mathcal{G}$ is guaranteed to be bounded by augmenting the state space with a virtual state ω such that $\rho_{(s,a)}(\omega) = 1$ for all (s,a) in $\mathcal{G} \times \mathcal{A}$, and $r = 0$ after reaching ω. Here a policy that is guaranteed to eventually reach $\mathcal{G}$ is called a *proper policy* (James & Collins, 2006; Van Niekerk *et al.*, 2019). To guarantee that an optimal policy does reach an absorbing state, it is assumed that the value functions for improper policies—those that never reach absorbing states—are unbounded below.

For the EVF of a deterministic shortest path task, the choice of $\bar{r}_{\text{MIN}}$ is particularly meaningful. It can be shown that for $\bar{r}_{\text{MIN}} \leq \min\{r_{\text{MIN}}, (r_{\text{MIN}} - r_{\text{MAX}})D\}$,[1] where D is the diameter of the MDP (Jaksch *et al.*, 2010), the EVF encodes how to achieve any goal in an environment irrespective of the specific task from which it was learned. We show this in Theorem 4.2.2. This choice of $\bar{r}_{\text{MIN}}$ says that since deterministic shortest path tasks

[1]The diameter is defined as $D = \max_{s \neq s' \in \mathcal{S}} \min_\pi \mathbb{E}\left[T(s'|\pi, s)\right]$, where T is the number of timesteps required to first reach s' from s under π.

require an agent to learn to achieve desired terminal states, if the agent enters a terminal state for a *different* task, it should receive the largest penalty possible.

Henceforth we will let $\bar{r}_{\text{MIN}} \leq \min\{r_{\text{MIN}}, (r_{\text{MIN}} - r_{\text{MAX}})D\}$ for deterministic shortest path tasks.[2]

Theorem 4.2.2. *Denote $\mathcal{S}^- = \mathcal{S} \setminus \mathcal{G}$ as the non-terminal states of a set of deterministic shortest path tasks $\mathcal{M}$. Let $M_1, M_2 \in \mathcal{M}$, and let each g in $\mathcal{G}$ define MDPs $M_{1,g}$ and $M_{2,g}$ with reward functions*

$$r_{M_{1,g}}(s,a) := \bar{r}_{M_1}(s,g,a) \text{ and } r_{M_{2,g}}(s,a) := \bar{r}_{M_2}(s,g,a) \text{ for all } (s,a) \text{ in } \mathcal{S} \times \mathcal{A}.$$

Then for all g in $\mathcal{G}$ and s in $\mathcal{S}^-$,

$$\pi_g^*(s) \in \arg\max_{a \in \mathcal{A}} Q^*_{M_{1,g}}(s,a) \text{ iff } \pi_g^*(s) \in \arg\max_{a \in \mathcal{A}} Q^*_{M_{2,g}}(s,a).$$

Proof. Let $g \in \mathcal{G}, s \in \mathcal{S}^-$ and let π_g^* be defined by

$$\pi_g^*(s') \in \arg\max_{a \in \mathcal{A}} Q^*_{M_1,g}(s,a) \text{ for all } s' \in \mathcal{S}.$$

If g is unreachable from s, then we are done since for all (s', a) in $\mathcal{S} \times \mathcal{A}$ we have

$$\begin{aligned} g \neq s' &\implies r_{M_{1,g}}(s',a) = \begin{cases} \bar{r}_{\text{MIN}}, & \text{if } s' \in \mathcal{G} \\ r_{s',a}, & \text{otherwise} \end{cases} = r_{M_{2,g}}(s',a) \\ &\implies M_{1,g} = M_{2,g}. \end{aligned}$$

If g *is* reachable from s, then we show that following π_g^* must reach g. Since π_g^* is proper, it must reach a terminal state $g' \in \mathcal{G}$. Assume $g' \neq g$. Let π_g be a policy that produces the shortest trajectory to g. Let $G^{\pi_g^*}$ and G^{π_g} be the returns for the respective policies. Then,

$$\begin{aligned}
&G^{\pi_g^*} \geq G^{\pi_g} \\
&\quad\implies G^{\pi_g^*}_{T-1} + r_{M_{1,g}}(g', \pi_g^*(g')) \geq G^{\pi_g}, \\
&\quad\text{where } G^{\pi_g^*}_{T-1} = \sum_{t=0}^{T-1} r_{M_{1,g}}(s_t, \pi_g^*(s_t)) \text{ and } T \text{ is the time at which } g' \text{ is reached.} \\
&\quad\implies G^{\pi_g^*}_{T-1} + \bar{r}_{\text{MIN}} \geq G^{\pi_g}, \text{ since } g \neq g' \in \mathcal{G} \\
&\quad\implies \bar{r}_{\text{MIN}} \geq G^{\pi_g} - G^{\pi_g^*}_{T-1} \\
&\quad\implies (r_{\text{MIN}} - r_{\text{MAX}})D \geq G^{\pi_g} - G^{\pi_g^*}_{T-1}, \text{ by definition of } \bar{r}_{\text{MIN}} \\
&\quad\implies G^{\pi_g^*}_{T-1} - r_{\text{MAX}}D \geq G^{\pi_g} - r_{\text{MIN}}D, \text{ since } G^{\pi_g} \geq r_{\text{MIN}}D \\
&\quad\implies G^{\pi_g^*}_{T-1} - r_{\text{MAX}}D \geq 0 \\
&\quad\implies G^{\pi_g^*}_{T-1} \geq r_{\text{MAX}}D.
\end{aligned}$$

But this is a contradiction since the result obtained by following an optimal trajectory up to a terminal state without the reward for entering the terminal state must be strictly

[2]In practice, we can simply set $\bar{r}_{\text{MIN}}$ to be the lowest finite value representable by the data type used for rewards.

less that receiving r_{MAX} for every step of the longest possible optimal trajectory. Hence we must have $g' = g$. Similarly, all optimal policies of $M_{2,g}$ must reach g. Hence $\pi^*_g(s) \in \arg\max_{a\in\mathcal{A}} Q^*_{M_{2,g}}(s,a)$. Since M_1 and M_2 are arbitrary elements of $\mathcal{M}$, the reverse implication holds too.

□

Combining Theorems 4.2.1 and 4.2.2, we can extract the greedy action from the extended value function by first maximising over goals, and then selecting the maximising action: $\pi^*(s) \in \arg\max_{a\in\mathcal{A}} \max_{g\in\mathcal{G}} \bar{Q}^*(s,g,a)$. If we consider the extended value function to be a set of standard value functions (one for each goal), then this is equivalent to first performing generalised policy improvement (Barreto *et al.*, 2017), and then selecting the greedy action.

Finally, much like the regular definition of value functions, the extended Q-value function can be written as the sum of rewards received by the agent until first encountering a terminal state.

Corollary 4.2.2.1. *Denote $G^*_{s:g,a} \in \mathbb{R}$ as the sum of rewards starting from s, taking the action a, and following an optimal policy up until, but not including, g. Then let $M \in \mathcal{M}$ and $\bar{Q}^*_M$ be the extended Q-value function. Then for all $s \in \mathcal{S}, g \in \mathcal{G}, a \in \mathcal{A}$, we have that*

$$\bar{Q}^*_M(s,g,a) = G^*_{s:g,a} + \bar{r}_M(s',g,a'), \text{ where } s' \in \mathcal{G} \text{ and } a' \in \arg\max_{b\in\mathcal{A}} \bar{r}_M(s',g,b).$$

Proof. This follows directly from Theorem 4.2.2. Since all tasks $M \in \mathcal{M}$ share the same optimal policy π^*_g up to (but not including) the goal state $g \in \mathcal{G}$, their return $G^{\pi^*_g}_{T-1} = \sum_{t=0}^{T-1} r_M(s_t, \pi^*_g(s_t))$ is the same up to (but not including) g. □

4.2.3 EVFs for Discounted Goal Reaching Tasks

Another class of MDPs that naturally model goal reaching tasks is discounted MDPs with an absorbing set $\mathcal{G} \subseteq \mathcal{S}$ and zero internal rewards (Abel *et al.*, 2018). Since these MDPs are discounted, having no internal rewards and positive goal rewards ensures the agent learns to reach the absorbing states (if they are indeed reachable and the discounting is sufficiently small). Hence this class of MDPs have reward functions of the form $r : \mathcal{S} \times \mathcal{A} \mapsto [0, r_{\text{MAX}}]$.

A natural choice for $\bar{r}_{\text{MIN}}$ here is 0. This gives us similar results to Theorem 4.2.2 and Corollary 4.2.2.1. Henceforth we will let $\bar{r}_{\text{MIN}} = r_{\text{MIN}} = 0$ for discounted goal-reaching tasks.

Theorem 4.2.3. *Denote $\mathcal{S}^- = \mathcal{S} \setminus \mathcal{G}$ as the non-terminal states of a set of discounted goal-reaching tasks $\mathcal{M}$. Let $M_1, M_2 \in \mathcal{M}$, and let each g in $\mathcal{G}$ define MDPs $M_{1,g}$ and $M_{2,g}$ with reward functions*

$$r_{M_{1,g}}(s,a) := \bar{r}_{M_1}(s,g,a) \text{ and } r_{M_{2,g}}(s,a) := \bar{r}_{M_2}(s,g,a) \text{ for all } (s,a) \text{ in } \mathcal{S} \times \mathcal{A}.$$

Then for all g in $\mathcal{G}$ and s in $\mathcal{S}^-$, there exist an optimal policy π^*_g such that

$$\pi^*_g(s) \in \arg\max_{a\in\mathcal{A}} Q^*_{M_{1,g}}(s,a) \text{ and } \pi^*_g(s) \in \arg\max_{a\in\mathcal{A}} Q^*_{M_{2,g}}(s,a).$$

Proof. Let $g \in \mathcal{G}, s \in \mathcal{S}^-$.

If g is unreachable from s, then we are done since for all (s', a) in $\mathcal{S} \times \mathcal{A}$ we have

$$\begin{aligned} g \neq s' &\implies r_{M_{1,g}}(s', a) = \begin{cases} \bar{r}_{\text{MIN}}, & \text{if } s' \in \mathcal{G} \\ r_{s',a}, & \text{otherwise} \end{cases} = r_{M_{2,g}}(s', a) \\ &\implies M_{1,g} = M_{2,g}. \end{aligned}$$

If g *is* reachable from s, then there exist a π^*_g given by

$$\pi^*_g(s') \in \arg\max_{a \in \mathcal{A}} Q^*_{M_{1,g}}(s, a) \text{ for all } s' \in \mathcal{S}.$$

that reaches g. This follows from the fact that action values are bounded below by 0 (since rewards are non-negative) and any policy π_g that doesn't reach g leads to $Q^{\pi_g}_{M_{1,g}}(s, a) = 0$ (since $r_{M_{1,g}}(s, a) = r_{s,a} = 0$ at internal states and $r_{M_{1,g}}(s, a) = \bar{r}_{\text{MIN}} = 0$ at absorbing states $s \neq g$). Similarly, there exist an optimal policy of $M_{2,g}$ that reaches g. Hence $\pi^*_g(s) \in \arg\max_{a \in \mathcal{A}} Q^*_{M_{2,g}}(s, a)$.

□

Corollary 4.2.3.1. *Denote $T^*_{s:g,a}$ as the length of the trajectory generated by an optimal policy π^*_g starting from s and taking action a. Then let $M \in \mathcal{M}$ and $\bar{Q}^*_M$ be the extended Q-value function. Then for all $s \in \mathcal{S}, g \in \mathcal{G}, a \in \mathcal{A}$, there exists a $T^*_{s:g,a} \in \mathbb{R} \cup \{\infty\}$ such that*

$$\bar{Q}^*_M(s, g, a) = \gamma^{T^*_{s:g,a}} r_M(g, a'), \ \textit{where } a' \in \arg\max_{b \in \mathcal{A}} r_M(g, b).$$

Proof. This follows directly from Theorem 4.2.3. Since all tasks $M \in \mathcal{M}$ share a common optimal policy π^*_g up to (but not including) g, the length $T^*_{s:g,a}$ of the trajectory generated by it is the same. Further more since $\bar{r}_M(s, g, a) = r_{s,a} = 0$ at internal states and $\bar{r}_M(s, g, a) = \bar{r}_{\text{MIN}} = 0$ at absorbing states $s \neq g$, the action values only depend on the rewards at $s = g$ where $\bar{r}_M(s, g, a) = r_M(g, a)$. □

4.3 Learning EVFs

Having formally established EVFs, we now propose ways of learning them. We note that while learning methods in the large body of work on goal conditioned value functions can be used to learn EVFs, they require reward functions to be available to the learning methods (Andrychowicz *et al.*, 2017; Schaul *et al.*, 2015; Kaelbling, 1993; Veeriah *et al.*, 2018; Foster & Dayan, 2002; Mirowski *et al.*, 2017; Moore *et al.*, 1999). Here we want to learn EVFs only with rewards obtained by interacting with the environment, just like in learning regular value functions.

Since EVFs can be thought of as a set of regular value functions each defined by a goal $g \in \mathcal{G}$, they can be learned by any suitable method in reinforcement learning, albeit not necessarily in an efficient way. Here we propose an extended version of Q-learning (Watkins, 1989) and deep Q-learning (Mnih *et al.*, 2015) to learn EVFs in the tabular and function-approximation case respectively.

4.3.1 Tabular Case

We use a modified version of Q-learning called *extended Q-learning* to learn the extended action-value functions described previously. Our algorithm differs in a number of ways from standard Q-learning: we keep track of the set of terminating states seen so far, and at each timestep we update the extended Q-value function with respect to both the current state and action, as well as all goals encountered so far. We also use the definition of the extended reward function, and so if the agent encounters a terminal state of a different task, it receives reward $\bar{r}_{\text{MIN}}$. The full pseudocode is shown in Figure 4.1.

```
Algorithm 1: Extended $Q$-learning
  Input: Learning rate $\alpha$, discount factor $\gamma$, exploration constant $\varepsilon$, lower-bound
         extended reward $\bar{r}_{\text{MIN}}$
  Initialise $\bar{Q} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ arbitrarily
  $\mathcal{G} \leftarrow \varnothing$
  while $\bar{Q}$ is not converged do
      Initialise state $s$
      while $s$ is not terminal do
          if $\mathcal{G} = \varnothing$ then
              Select random action $a$
          else
              $a \leftarrow \begin{cases} \arg\max_{a' \in \mathcal{A}} \left( \max_{g \in \mathcal{G}} \bar{Q}(s, g, a') \right) & \text{with probability } 1 - \varepsilon \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$
          end
          Choose $a$ from $s$ according to policy derived from $\bar{Q}$
          Take action $a$, observe $r$ and $s'$
          foreach $g \in \mathcal{G}$ do
              if $s'$ is terminal then
                  if $s' \neq g$ then
                      $\delta \leftarrow \bar{r}_{\text{MIN}} - \bar{Q}(s, g, a)$
                  else
                      $\delta \leftarrow r - \bar{Q}(s, g, a)$
                  end
              else
                  $\delta \leftarrow r + \gamma \max_b \bar{Q}(s', g, b) - \bar{Q}(s, g, a)$
              end
              $\bar{Q}(s, g, a) \leftarrow \bar{Q}(s, g, a) + \alpha\delta$
          end
          $s \leftarrow s'$
      end
      $\mathcal{G} \leftarrow \mathcal{G} \cup \{s\}$
  end
  return $\bar{Q}$
```

Figure 4.1: A Q-learning algorithm for learning extended value functions. Note that the greedy action selection step is equivalent to generalised policy improvement (Barreto *et al.*, 2017) over the set of extended value functions.

4.3.2 Function Approximation Case

Similarly to the modifications done on Q-learning to get extended Q-learning, we modify deep Q-learning to get *extended deep Q-learning.* Our approach differs in that the neural network—referred to as a Deep Q-Network (DQN) (Mnih *et al.*, 2015)—takes a goal state as additional input. Additionally, when a terminal state is encountered, it is added to the collection of goals seen so far, and when learning updates occur, these goals are sampled randomly from a goal buffer. The full pseudocode is shown in Figure 4.2.

```
Algorithm 2: Extended Deep $Q$-learning
  Input: Maximum timesteps $T$, Target network updates $C$, Learning rate $\alpha$,
         discount factor $\gamma$, exploration constant $\varepsilon$, lower-bound extended reward
         $\bar{r}_{\text{MIN}}$
  Initialize network $\bar{Q}_\theta : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ with random weights $\theta$
  Initialize target network $\bar{Q}'_{\theta'} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ with random weights $\theta'$
  Initialize goal and replay buffers $\mathcal{G} \leftarrow \varnothing$ and $\mathcal{D} \leftarrow \varnothing$ respectively
  $t \leftarrow 0$
  while $t \leq T$ do
      Initialise state $s$
      while $s$ is not terminal do
          if $\mathcal{G} = \varnothing$ then
              Select random action $a$
          else
              Select action $a$ using $\epsilon$-greedy generalised policy improvement on $\bar{Q}_\theta$
          end
          Choose $a$ from $s$ according to policy derived from $\bar{Q}_\theta$
          Take action $a$, observe $r$ and $s'$
          $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s, a, r, s')\}$
          Sample mini-batch of goals $\mathcal{G}' \subseteq \mathcal{G}$ and transitions $\mathcal{D}' \subseteq \mathcal{D}$
          foreach $g, (s, a, r, s') \in \mathcal{G}' \times \mathcal{D}'$ do
              if $s'$ is terminal then
                  $\bar{r} \leftarrow \bar{r}_{\text{MIN}}$ if $s' \neq g$ else $r$
                  $\delta \leftarrow \bar{r} - \bar{Q}'_{\theta'}(s, g, a)$
              else
                  $\delta \leftarrow r + \gamma \max_b \bar{Q}_\theta(s', g, b) - \bar{Q}'_{\theta'}(s, g, a)$
              end
              Perform a gradient descent step on $\delta^2$
          end
          Every $C$ steps update target network $\theta' \leftarrow \theta$
          $s \leftarrow s'$ ; $t \leftarrow t + 1$
      end
      $\mathcal{G} \leftarrow \mathcal{G} \cup \{s\}$
  end
  return $Q$
```

Figure 4.2: A deep Q-learning algorithm for learning extended value functions with function approximation. The ϵ-greedy action selection using generalised policy improvement is done similarly to Algorithm 1.

4.4 Experiments

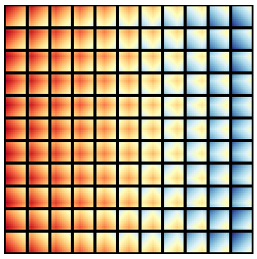
In this section we learn extended value functions for both deterministic shortest path tasks and discounted goal-reaching tasks. We observe the proven properties from the previous sections.

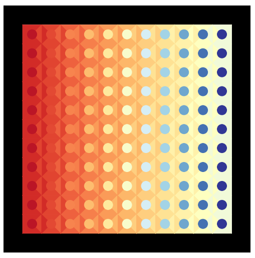
4.4.1 Tabular Case

Consider the simple grid world domain introduced in Example 3.4.1 where an agent may be asked to go to any position on the xy-plane. The agent learns deterministic shortest path tasks and discounted goal-reaching tasks, where

- Deterministic shortest path tasks: The internal rewards (rewards for all non-terminal states) are -0.1 and the goal rewards (rewards on the absorbing set) range from -0.1 to 2.
- Discounted goal-reaching tasks: The internal rewards (rewards for all non-terminal states) are 0 and the goal rewards (rewards on the absorbing set) range from 0 to 2. The discount factor used is $\gamma = 0.95$.

We train an agent on the task M_{L}, in which the agent must learn to navigate to the left of the xy-plane. Figures 4.3(a) and 4.4(a) shows the learned EVFs for both deterministic shortest path and discounted goal reaching cases respectively. The figures are generated by plotting the value functions for each goal state, and displaying it at the position of the respective goal states. We observe that the learned EVFs do indeed encode how to achieve multiple goals in the environment. Notice how for the goal positions next to the right wall (the rightmost column of the plot), the EVF for the discounted goal reaching task has zero values everywhere since the agent receives no reward at those goals.


(a) Extended state-value function obtained by acting greedily over actions on extended action-value function.


(b) Action-value function obtained by acting greedily over goals on extended action-value function.

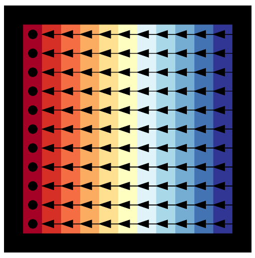
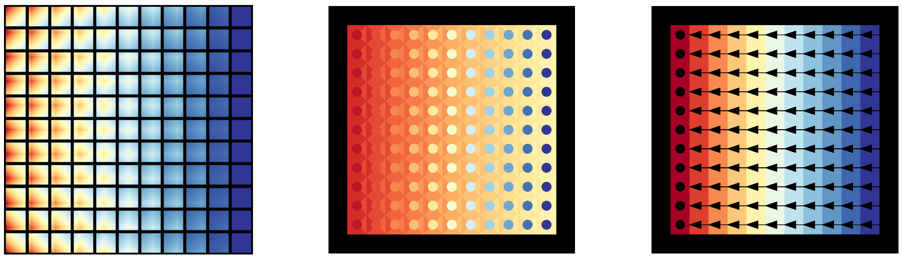

(c) State-value function and policy obtained by acting greedily over action-value function.

Figure 4.3: Learned EVF for the deterministic shortest path task of going to the left of the gridworld. (a) Each square shows the value of all internal states with respect to the goal state at that position. (b) Each square is divided into 5 parts representing the values for each action at that position. The circles show the value of choosing the stay action. (c) Arrows represent the optimal action in a given state.

Note how we also recover the optimal action-value function, state-value function, and policy for both types of goal oriented tasks. This is shown for the deterministic shortest



(a) Extended state-value function obtained by acting greedily over actions on extended action-value function.

(b) Action-value function obtained by acting greedily over goals on extended action-value function.

(c) State-value function and policy obtained by acting greedily over action-value function.

Figure 4.4: Learned EVF for the discounted goal reaching task of going to the left of the gridworld. (a) Each square shows the value of all internal states with respect to the goal state at that position. (b) Each square is divided into 5 parts representing the values for each action at that position. The circles show the value of choosing the stay action. (c) Arrows represent the optimal action in a given state.

path and discounted goal reaching cases in Figures 4.3(b)-(c) and 4.4(b)-(c) respectively. It demonstrates the results proven in Section 4.2.

By learning extended value functions, an agent learns a massive number of diverse solutions to a single task. However, the upfront cost of learning is likely to be higher since we must learn not only the optimal Q function, but the optimal Q function with respect to every goal. We investigate how the sample complexity of learning extended value functions and learning standard value functions scales with the number of tasks (Figure 4.5). We compare extended Q-learning to Q-learning, since extended Q-learning is the method we propose for learning extended value functions. We observe that the number of samples required to learn the extended value function is greater than learning a standard value function. However, both scale linearly and differ only by a constant factor. We will see in the next chapter how we can leverage the extended value functions to improve transfer in a multi-task setting, which amortises the upfront cost over multiple tasks.

4.4.2 Function Approximation Case

Finally, we demonstrate that our modified learning method can also be used to learn high-dimensional domains where function approximation is required.[3] In our experiments, we use a DQN with the following architecture:

1. Three convolutional layers:
   (a) Layer 1 has 6 input channels, 32 output channels, a kernel size of 8 and a stride of 4.
   (b) Layer 2 has 32 input channels, 64 output channels, a kernel size of 4 and a stride of 2.
   (c) Layer 3 has 64 input channels, 64 output channels, a kernel size of 3 and a stride of 1.

[3]See Section 4.3.2 for a description of the learning method.

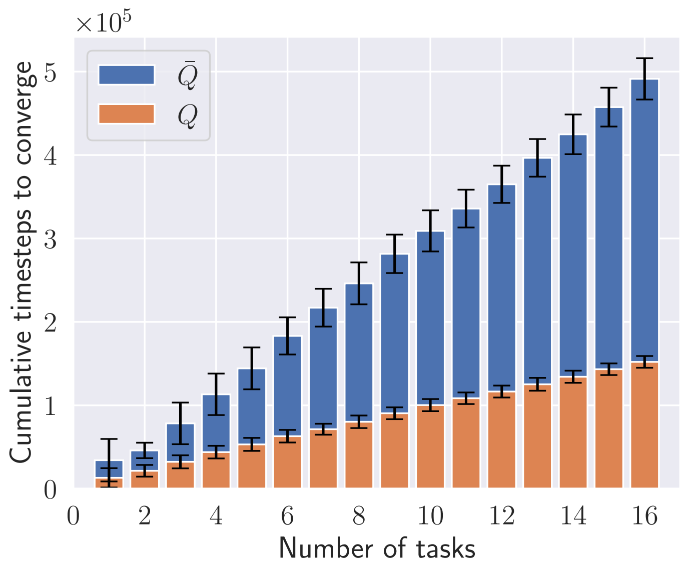


Figure 4.5: Cumulative number of samples required to learn optimal extended and regular value functions. Error bars represent standard deviations over 100 runs.

2. Two fully-connected linear layers:
   (a) Layer 1 has input size 3136 and output size 512 and uses a ReLU activation function.
   (b) Layer 2 has input size 512 and output size 5 with no activation function.

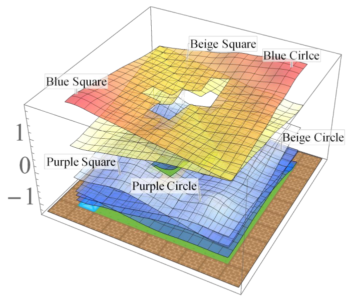
We used the ADAM optimiser with mini-batch size 32 and a learning rate of 10^{-4}. We trained every 4 timesteps and update the target Q-network every 1000 steps. Finally, we used ϵ-greedy exploration, annealing ϵ to 0.01 over 100000 timesteps.

We use the same video game environment as Van Niekerk *et al.* (2019), where an agent must navigate a 2D world and collect objects of different shapes and colours. The state space is an 84×84 RGB image, and the agent is able to move in any of the four cardinal directions. The agent also possesses a `pick-up` action, which allows it to collect an object when standing on top of it. There are two shapes (squares and circles) and three colours (blue, beige and purple) for a total of six unique objects. The position of the agent is randomised at the start of each episode.

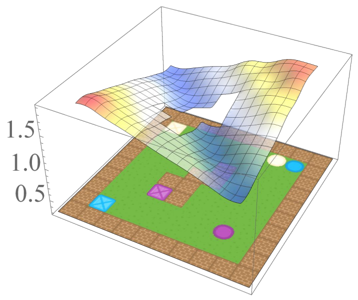
We train the agent on the blue task, in which it needs to optimally to navigate to blue objects and collect them. Figures 4.6 and 4.7 shows the learned EVFs and the result of maximising over goals for both discounted and deterministic shortest path cases respectively. To generate the EVFs, we place the agent at every location and compute the maximum output of the network over all actions for each goal. We then interpolate between the points to smooth the graph. Any error in the visualisation is due to the use of non-linear function approximation.

4.5 Conclusion

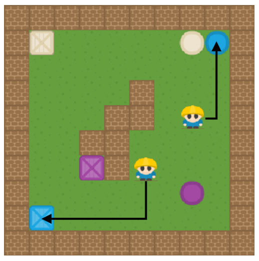
For an agent to be able to solve new tasks optimally in an environment without extra learning, it needs to have gained sufficient information from its experience when learning to solve previous tasks. Its objective during learning hence shouldn't just be how to



(a) Extended state-value function obtained by maximizing over actions on extended action-value function.

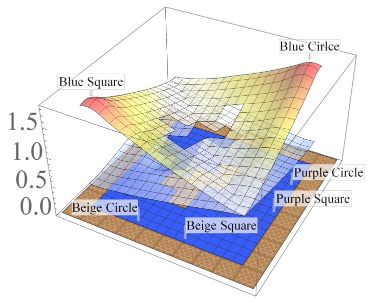


(b) Value function obtained by maximizing over goals on extended state-value function.

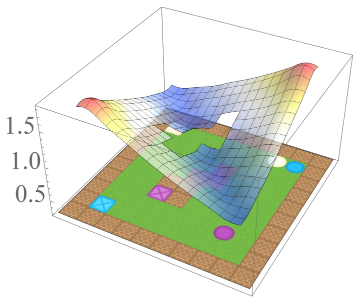


(c) Sample trajectories of agent solving the task.

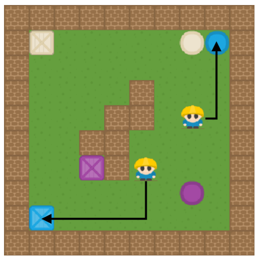
Figure 4.6: Learned EVF for the deterministic shortest path task of collecting blue objects in the video game domain. The EVF shows the value function with respect to each goal plotted on the same axis.



(a) Extended state-value function obtained by maximizing over actions on extended action-value function.



(b) Value function obtained by maximizing over goals on extended state-value function.



(c) Sample trajectories of agent solving the task.

Figure 4.7: Learned EVF for the discounted goal reaching task of collecting blue objects in the video game domain. The EVF shows the value function with respect to each goal plotted on the same axis.

optimally solve the current task, but how to gain as much knowledge as possible from its experience while doing so. We introduced a new type of value function that captures this idea. As the agent acts in the environment it achieves outcomes that are good or bad to various degree. Rather than focusing on just the best outcomes and how to achieve them, it learns how to achieve any possible outcome. After training it knows about the various possible outcomes of the task and what behaviours lead to them. This is similar to the idea of *mastery* proposed by Veeriah *et al.* (2018). When trying to solve the task optimally, it is a simple matter for the agent to choose the best outcome. This is a much richer knowledge representation than standard value functions which learn just how to achieve the best outcomes. An agent may now leverage this rich knowledge about learned tasks to solve new tasks optimally. We investigate this in the next chapter.

Chapter 5

Composing Extended Value Functions

5.1 Introduction

In the previous chapters we formalised the logical composition of tasks, showed how standard value functions are insufficient to solve them zero-shot, and introduced EVFs as a richer type of value function that may encode sufficient information about learned tasks to solve new ones zero-shot. In this chapter we formally show that EVFs are indeed sufficient to solve logical composition of tasks zero-shot. We do this by first formalising their composition under the relevant algebraic structures, just as was done with task compositions. This gives us mathematical tools which can used to formally show zero-shot composition and further explore some additional properties of the established structures.

Hence the main contributions of this chapter are as follows:

- We introduce EVF Lattices, which gives a mathematical structure for doing arbitrary disjunctions and conjunctions of EVFs. It also gives a graphical representation for the space of EVFs.

- We define the negation of EVFs, and hence establish their logical compositions. This gives the ability to do arbitrary negations, disjunctions and conjunctions of EVFs.

- We prove zero-shot composition of tasks with dense goal rewards. This guarantees a compositional explosion of skills.

- We prove zero-shot composition of tasks with sparse rewards. This combined with the notion of base tasks guarantees a super-exponential explosion of skills.

- We prove that for tasks with sparse rewards, their EVF algebra is isomorphic to both their task algebra and the power set algebra of their goal space. It shows that given a set of desired goals, the specific isomorphism we consider can generate the EVF of the related task from just two EVFs. That is after learning just two EVFs, any task with sparse rewards can theoretically be solved without extra learning, regardless of the size of the task space.

- We show experimentally that zero-shot composition also holds in high dimensional environments that require function approximation.

- We show experimentally that zero-shot composition still holds even when the assumptions made for theoretical guarantees are relaxed.

This chapter is structured as follows. Section 5.2 formalises logical composition of EVFs under relevant algebraic structures. This is used to show zero-shot composition. Section 5.3 shows the isomorphism between tasks, EVFs and power set algebras. This is used to show how EVFs can be generated from two constant EVFs without extra learning. Section 5.4 demonstrates zero-shot composition in a video game domain. Finally, Section 5.5 investigates the effect of relaxing the assumptions made on the class of MDPs. It also investigates the trade off between the time it takes to learn EVFs versus the compositional explosion of skills obtained.

5.2 Algebra of EVFs

In this section we establish the logical composition of EVFs and show that this leads to zero-shot composition.

5.2.1 EVF Lattice

Similarly to how we formalised disjunction and conjunction of tasks using the lattice algebraic structure, we also use it to formalise the disjunction and conjunction of EVFs. Since EVFs are real valued functions, a natural partial order on them is pointwise $\leq$ (the usual $\leq$ on $\mathbb{R}$). We state the resulting poset formally as follows:

Proposition 5.2.1. *Let $\bar{\mathcal{Q}}^*$ be the set of optimal $\bar{Q}$-value functions for tasks in $\mathcal{M}$. Then $(\bar{\mathcal{Q}}^*, \leq)$ is a partially ordered set with the relation $\leq$ given by*

$$\bar{Q}_1^* \leq \bar{Q}_2^* \text{ if } \bar{Q}_1^*(s,g,a) \leq \bar{Q}_2^*(s,g,a) \text{ for all } (s,g,a) \in \mathcal{S} \times \mathcal{G} \times \mathcal{A}.$$

Proof. Follows from the usual $\leq$ relation on $\mathbb{R}$. □

While in general a partial order does not necessarily induce a lattice, it does for the poset of EVFs in which we are interested. This follows from the partial order on the set of EVFs given by the pointwise $\leq$. Since the EVFs are real valued, every set of points has a least upper bound (the max) and a greatest lower bound (the min). We show that the poset of EVFs of deterministic shortest path tasks and discounted goal reaching tasks have a supremum and an infimum for any pair of elements.

Proposition 5.2.2. *Let $\mathcal{M}$ be a set of deterministic shortest path tasks or a set of discounted goal reaching tasks. Let $\bar{\mathcal{Q}}^*$ be the set of optimal $\bar{Q}$-value functions for tasks in $\mathcal{M}$. Then for all $\bar{Q}_1^*, \bar{Q}_2^* \in \bar{\mathcal{Q}}^*$,*

(i): $\sup\{\bar{Q}_1^*, \bar{Q}_2^*\} = \bar{Q}_{sup}^* \in \bar{\mathcal{Q}}^*$, *where* $\bar{Q}_{sup}^* : (s,g,a) \mapsto \sup\{\bar{Q}_1^*(s,g,a), \bar{Q}_2^*(s,g,a)\}$.

(ii): $\inf\{\bar{Q}_1^*, \bar{Q}_2^*\} = \bar{Q}_{inf}^* \in \bar{\mathcal{Q}}^*$, *where* $\bar{Q}_{inf}^* : (s,g,a) \mapsto \inf\{\bar{Q}_1^*(s,g,a), \bar{Q}_2^*(s,g,a)\}$.

Proof. Let $M_1, M_2 \in \mathcal{M}$. Then for all (s,g,a) in $\mathcal{S} \times \mathcal{G} \times \mathcal{A}$,

(i): If $\mathcal{M}$ is a set of deterministic shortest path tasks, then

$$\begin{aligned}
\bar{Q}^*_{sup}(s,g,a) &= \sup\{\bar{Q}^*_{M_1}(s,g,a), \bar{Q}^*_{M_2}(s,g,a)\} \\
&= \sup\{G^*_{s:g,a} + \bar{r}_{M_1}(s',g,a'), G^*_{s:g,a} + \bar{r}_{M_2}(s',g,a'')\} \quad \textit{(Corollary 4.2.2.1)} \\
&= G^*_{s:g,a} + \sup\{\bar{r}_{M_1}(s',g,a'), \bar{r}_{M_2}(s',g,a'')\} \\
&= G^*_{s:g,a} + \bar{r}_{M_1 \vee M_2}(s',g,a') \\
&= \bar{Q}^*_{M_1 \vee M_2}(s,g,a) \\
\implies \bar{Q}^*_{sup} &= \bar{Q}^*_{M_1 \vee M_2} \in \bar{\mathcal{Q}}^*.
\end{aligned}$$

If $\mathcal{M}$ is a set of discounted goal reaching tasks, then

$$\begin{aligned}
\bar{Q}^*_{sup}(s,g,a) &= \sup\{\bar{Q}^*_{M_1}(s,g,a), \bar{Q}^*_{M_2}(s,g,a)\} \\
&= \sup\{\gamma^{T^*_{s:g,a}} r_{M_1}(g,a'), \gamma^{T^*_{s:g,a}} r_{M_2}(g,a')\} \quad \textit{(Corollary 4.2.3.1)} \\
&= \gamma^{T^*_{s:g,a}} \sup\{r_{M_1}(g,a'), r_{M_2}(g,a')\} \quad \textit{(since } \gamma \textit{ is non-negative)} \\
&= \gamma^{T^*_{s:g,a}} r_{M_1 \vee M_2}(g,a') \\
&= \bar{Q}^*_{M_1 \vee M_2}(s,g,a) \\
\implies \bar{Q}^*_{sup} &= \bar{Q}^*_{M_1 \vee M_2} \in \bar{\mathcal{Q}}^*.
\end{aligned}$$

Since $\bar{Q}^*_{sup}$ is in $\bar{\mathcal{Q}}^*$, it follows from the pointwise $\leq$ on $\mathbb{R}$ that it is the lowest upper bound of $\bar{Q}^*_1$ and $\bar{Q}^*_2$.

(ii): Follows similarly to (i).

□

This means that the set of EVFs $\bar{\mathcal{Q}}^*$ of deterministic shortest path tasks or discounted goal reaching tasks forms a lattice $(\bar{\mathcal{Q}}^*, \vee, \wedge)$ with $\vee$ and $\wedge$ given by $\bar{Q}^*_1 \vee \bar{Q}^*_2 := \sup\{\bar{Q}^*_1, \bar{Q}^*_2\}$ and $\bar{Q}^*_1 \wedge \bar{Q}^*_2 := \inf\{\bar{Q}^*_1, \bar{Q}^*_2\}$ respectively. We define these formally as follows:

Definition 5.2.1. *Let $\mathcal{M}$ be a set of deterministic shortest path tasks or a set of discounted goal reaching tasks. Let $\bar{\mathcal{Q}}^*$ be the set of optimal $\bar{Q}$-value functions for tasks in $\mathcal{M}$. The join and meet operators on $\bar{\mathcal{Q}}^*$ are respectively given by*

$$\begin{aligned}
\vee : \ & \bar{\mathcal{Q}}^* \times \bar{\mathcal{Q}}^* \to \bar{\mathcal{Q}}^* \\
& (\bar{Q}^*_1, \bar{Q}^*_2) \mapsto \bar{Q}^*_1 \vee \bar{Q}^*_2, \ \textit{where } \bar{Q}^*_1 \vee \bar{Q}^*_2 : \ \mathcal{S} \times \mathcal{G} \times \mathcal{A} \to \mathbb{R} \\
& \qquad\qquad\qquad\qquad\qquad\qquad (s,g,a) \mapsto \sup\{\bar{Q}^*_1(s,g,a), \bar{Q}^*_2(s,g,a)\},
\end{aligned}$$

$$\begin{aligned}
\wedge : \ & \bar{\mathcal{Q}}^* \times \bar{\mathcal{Q}}^* \to \bar{\mathcal{Q}}^* \\
& (\bar{Q}^*_1, \bar{Q}^*_2) \mapsto \bar{Q}^*_1 \wedge \bar{Q}^*_2, \ \textit{where } \bar{Q}^*_1 \wedge \bar{Q}^*_2 : \ \mathcal{S} \times \mathcal{G} \times \mathcal{A} \to \mathbb{R} \\
& \qquad\qquad\qquad\qquad\qquad\qquad (s,g,a) \mapsto \inf\{\bar{Q}^*_1(s,g,a), \bar{Q}^*_2(s,g,a)\}.
\end{aligned}$$

In fact $(\bar{\mathcal{Q}}^*, \vee, \wedge)$ forms a distributive lattice. We state this as follows:

Proposition 5.2.3. *Let $\mathcal{M}$ be a set of deterministic shortest path tasks or a set of discounted goal-reaching tasks. Let $\bar{\mathcal{Q}}^*$ be the set of optimal $\bar{Q}$-value functions for tasks in $\mathcal{M}$. Then $(\bar{\mathcal{Q}}^*, \vee, \wedge)$ is a distributive lattice.*

Proof. Follows from the distributivity of inf and sup. □

Given a non-empty finite set $\mathcal{O}$ of lower bounded subsets of EVFs $\mathcal{N} \subset \bar{\mathcal{Q}}^*$, the EVF lattice $(\bar{\mathcal{Q}}^*, \vee, \wedge)$ gives us the principled way of specifying the disjunction of conjunctions,

$$\bigvee_{\mathcal{N}\in\mathcal{O}} \left(\bigwedge_{N\in\mathcal{N}} N \right)(s,g,a) = \sup_{\mathcal{N}\in\mathcal{O}} \left(\inf_{N\in\mathcal{N}} \bar{Q}^*_N(s,g,a) \right).$$

Similarly, given a non-empty finite set $\mathcal{O}$ of upper bounded subsets of EVFs $\mathcal{N} \subset \bar{\mathcal{Q}}^*$, the conjunction of disjunctions is given by,

$$\bigwedge_{\mathcal{N}\in\mathcal{O}} \left(\bigvee_{N\in\mathcal{N}} N \right)(s,g,a) = \inf_{\mathcal{N}\in\mathcal{O}} \left(\sup_{N\in\mathcal{N}} \bar{Q}^*_N(s,g,a) \right).$$

If $\bar{\mathcal{Q}}^*$ is bounded, then it forms a complete distributive lattice. This means that the above will hold true for any non-empty subset of $\bar{\mathcal{Q}}^*$.

Having established a lattice algebra over tasks and extended value functions, we show that there exists an equivalence between the two. As a result, if we can write down a task under the lattice algebra, we can immediately write down the optimal value function for the task.

Theorem 5.2.1. *Let $\mathcal{M}$ be a set of deterministic shortest path tasks or a set of discounted goal-reaching tasks. Let $\bar{\mathcal{Q}}^*$ be the set of optimal $\bar{Q}$-value functions for tasks in $\mathcal{M}$. Let $\mathcal{F} : \mathcal{M} \to \bar{\mathcal{Q}}^*$ be any map from $\mathcal{M}$ to $\bar{\mathcal{Q}}^*$ such that $\mathcal{F}(M) = \bar{Q}^*_M$ for all M in $\mathcal{M}$. Then $\mathcal{F}$ is a homomorphism between the task lattice $(\mathcal{M}, \vee, \wedge)$ and the EVFs lattice $(\bar{\mathcal{Q}}^*, \vee, \wedge)$.*

Proof. Follows from the proofs of Propositions 5.2.2, which give

$$\bar{Q}^*_{M_1} \vee \bar{Q}^*_{M_2} = \sup\{\bar{Q}^*_{M_1}, \bar{Q}^*_{M_2}\} = \bar{Q}^*_{M_1 \vee M_2}$$

and

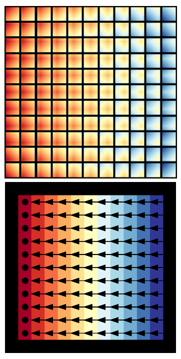
$$\bar{Q}^*_{M_1} \wedge \bar{Q}^*_{M_2} = \inf\{\bar{Q}^*_{M_1}, \bar{Q}^*_{M_2}\} = \bar{Q}^*_{M_1 \wedge M_2}$$

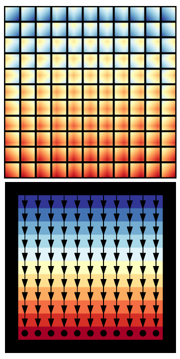
$\forall M_1, M_2 \in \mathcal{M}$. □

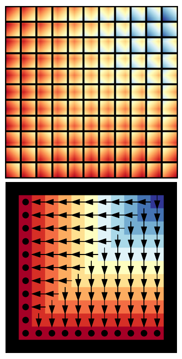
Experiment 5.2.1. *Consider the simple grid world domain introduced in Example 3.4.1 where an agent may be asked to go to any position on the xy-plane. The agent learns deterministic shortest path and discounted goal-reaching tasks, where*

- *Deterministic shortest path tasks: The internal rewards are -0.1 and the goal rewards range from -0.1 to 2.*
- *Discounted goal-reaching tasks: The internal rewards (rewards for all non-terminal states) are 0 and the goal rewards (rewards on the absorbing set) range from 0 to 2. The discounting used is $\gamma = 0.95$.*

We train an agent on the tasks M_L and M_D, in which the agent must learn to navigate to the left and bottom of the xy-plane respectively. Figure 5.1 shows the learned EVFs $\bar{Q}^*_L$ and $\bar{Q}^*_D$ together with their disjunction and conjunction for the deterministic shortest path case. Similarly Figure 5.2 shows these for the discounted goal reaching case. For each EVF figure, each plot shows the value function with respect to that goal position (the position of the plot). Finally, notice how the composition of EVFs exhibit the same semantics as those of rewards. This highlights the homomorphism proven in Theorem 5.2.1, showing the structural similarity between the task space and value function space.


(a) $\bar{Q}^*_{\text{L}}$


(b) $\bar{Q}^*_{\text{D}}$


(c) $\bar{Q}^*_{\text{L}} \vee \bar{Q}^*_{\text{D}}$

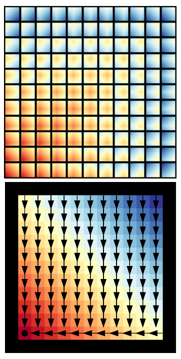
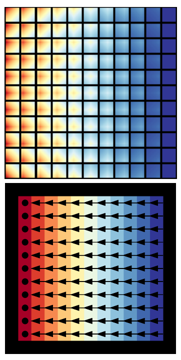
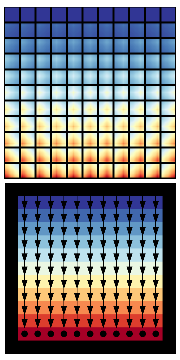
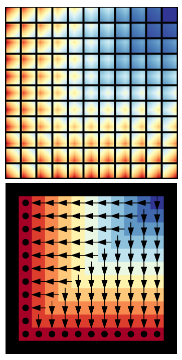

(d) $\bar{Q}^*_{\text{L}} \wedge \bar{Q}^*_{\text{D}}$

Figure 5.1: Showing the disjunction and conjunction of EVFs learned from deterministic shortest path tasks in gridworld domain. Figures (a)-(b) shows the EVFs for the left and down tasks. (c) shows the EVF of their disjunction. (d) shows the EVF of their conjunction.


(a) $\bar{Q}^*_{\text{L}}$


(b) $\bar{Q}^*_{\text{D}}$


(c) $\bar{Q}^*_{\text{L}} \vee \bar{Q}^*_{\text{D}}$

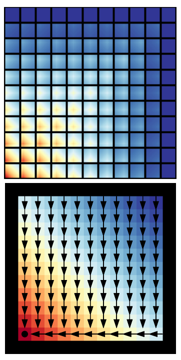

(d) $\bar{Q}^*_{\text{L}} \wedge \bar{Q}^*_{\text{D}}$

Figure 5.2: Showing the disjunction and conjunction of EVFs learned from discounted tasks in gridworld domain. Figures (a)-(b) shows the EVFs for the left and down tasks. (c) shows the EVF of their disjunction. (d) shows the EVF of their conjunction.

5.2.2 De Morgan EVF Algebra

Having formalised the meaning of disjunction and conjunction of EVFs, we next formalise the meaning of the negation of a EVFs. A De Morgan algebra enables us to define it by adding minimal required properties that encapsulates the desired semantics of a negation. We define the negation of an EVF as follows:

Definition 5.2.2. *Let $\bar{\mathcal{Q}}^*$ be the set of optimal $\bar{Q}$-value functions for tasks in a bounded set of tasks $\mathcal{M}$. Define $\bar{Q}^*_{INF}, \bar{Q}^*_{SUP} \in \bar{\mathcal{Q}}^*$ to be the optimal $\bar{Q}$-functions for the tasks $\mathcal{M}_{INF}, \mathcal{M}_{SUP} \in \mathcal{M}$. Then the negation operator is given by,*

$$\begin{aligned} \neg :\ & \bar{\mathcal{Q}}^* \to \bar{\mathcal{Q}}^* \\ & \bar{Q}^* \mapsto \neg\bar{Q}^*, \text{ where } \neg\bar{Q}^* :\ \mathcal{S} \times \mathcal{G} \times \mathcal{A} \to \mathbb{R} \\ & \qquad (s,g,a) \mapsto \left(\bar{Q}^*_{SUP}(s,g,a) + \bar{Q}^*_{INF}(s,g,a)\right) - \bar{Q}^*(s,g,a). \end{aligned}$$

We show that the negation of $\bar{Q}^* \in \bar{\mathcal{Q}}^*$ is indeed in $\bar{\mathcal{Q}}^*$ for the EVFs we are interested in.

Proposition 5.2.4. *Let $\mathcal{M}$ be a set of deterministic shortest path tasks or a set of discounted goal-reaching tasks satisfying Assumption 3.4.1. Let $\bar{\mathcal{Q}}^*$ be the set of optimal $\bar{Q}$-value functions for tasks in $\mathcal{M}$. Then $\neg\bar{Q}^* \in \bar{\mathcal{Q}}^*$ for all $\bar{Q}^* \in \bar{\mathcal{Q}}^*$.*

Proof. Let $M \in \mathcal{M}$. Then for all (s, g, a) in $\mathcal{S} \times \mathcal{G} \times \mathcal{A}$,

if $\mathcal{M}$ is a set of deterministic shortest path tasks, then

$$\begin{aligned}
\neg\bar{Q}^*_M(s,g,a) &= \left[\bar{Q}^*_{SUP}(s,g,a) + \bar{Q}^*_{INF}(s,g,a)\right] - \bar{Q}^*_M(s,g,a) \\
&= \left[(G^*_{s:g,a} + \bar{r}_{\mathcal{M}_{SUP}}(s',g,a')) + (G^*_{s:g,a} + \bar{r}_{\mathcal{M}_{INF}}(s',g,a'))\right] \\
&\quad - (G^*_{s:g,a} + \bar{r}_M(s',g,a')) \quad \text{(Corollary 4.2.2.1)} \\
&= G^*_{s:g,a} + (\bar{r}_{\mathcal{M}_{SUP}}(s',g,a') + \bar{r}_{\mathcal{M}_{INF}}(s',g,a')) - \bar{r}_M(s',g,a') \\
&= G^*_{s:g,a} + \bar{r}_{\neg M}(s',g,a') \\
&= \bar{Q}^*_{\neg M}(s,g,a).
\end{aligned}$$

if $\mathcal{M}$ is a set of discounted goal-reaching tasks, then

$$\begin{aligned}
\neg\bar{Q}^*_M(s,g,a) &= \left[\bar{Q}^*_{SUP}(s,g,a) + \bar{Q}^*_{INF}(s,g,a)\right] - \bar{Q}^*_M(s,g,a) \\
&= \left[(\gamma^{T^*_{s:g,a}} r_{\mathcal{M}_{SUP}}(g,a')) + (\gamma^{T^*_{s:g,a}} r_{\mathcal{M}_{INF}}(g,a'))\right] \\
&\quad - (\gamma^{T^*_{s:g,a}} r_M(g,a')) \quad \text{(Corollary 4.2.3.1)} \\
&= \gamma^{T^*_{s:g,a}}[(r_{\mathcal{M}_{SUP}}(g,a') + r_{\mathcal{M}_{INF}}(g,a')) - r_M(g,a')] \\
&= \gamma^{T^*_{s:g,a}} r_{\neg M}(g,a') \\
&= \bar{Q}^*_{\neg M}(s,g,a).
\end{aligned}$$

Hence $\neg\bar{Q}^*_M = \bar{Q}^*_{\neg M} \in \bar{\mathcal{Q}}^*$.

□

We now formalize the interaction of the negation of EVFs with the conjunction and disjunction of EVFs as follows:

Proposition 5.2.5. *Let $\mathcal{M}$ be a set of deterministic shortest path tasks or discounted goal-reaching tasks satisfying Assumption 3.4.1. Let $\bar{\mathcal{Q}}^*$ be the set of optimal $\bar{Q}$-value functions for tasks in $\mathcal{M}$. Then $(\bar{\mathcal{Q}}^*, \vee, \wedge, \neg, \bar{Q}^*_{SUP}, \bar{Q}^*_{INF})$ is a De Morgan algebra.*

Proof. Let $\bar{Q}^*_{M_1}, \bar{Q}^*_{M_2} \in \bar{\mathcal{Q}}^*$ be the optimal $\bar{Q}$-value functions for tasks $M_1, M_2 \in \mathcal{M}$ with reward functions r_{M_1} and r_{M_2}. We show that $\neg, \vee, \wedge$ satisfy the De Morgan algebra properties (i) – (vii).

(i)–(v): These follow from the properties of *inf* and *sup*.

(vi): This follows from the bounds $\bar{Q}^*_{SUP}, \bar{Q}^*_{INF} \in \bar{\mathcal{Q}}^*$ which are guaranteed to exist due to Assumption 3.4.1.

(vii): The first condition easily follows from the definition of $\neg$. For the second condition, we have that for all (s, g, a) in $\mathcal{S} \times \mathcal{G} \times \mathcal{A}$,

$$\begin{aligned}
\neg(\bar{Q}^*_{M_1} \vee \bar{Q}^*_{M_2})(s,g,a) &= (\bar{Q}^*_{SUP}(s,g,a) + \bar{Q}^*_{INF}(s,g,a)) - \sup_{M \in \{M_1,M_2\}} \bar{Q}^*_M(s,g,a) \\
&= (\bar{Q}^*_{SUP}(s,g,a) + \bar{Q}^*_{INF}(s,g,a)) + \inf_{M \in \{M_1,M_2\}} -\bar{Q}^*_M(s,g,a) \\
&= \inf_{M \in \{M_1,M_2\}} (\bar{Q}^*_{SUP}(s,g,a) + \bar{Q}^*_{INF}(s,g,a)) - \bar{Q}^*_M(s,g,a) \\
&= (\neg\bar{Q}^*_{M_1} \wedge \neg\bar{Q}^*_{M_2})(s,g,a).
\end{aligned}$$

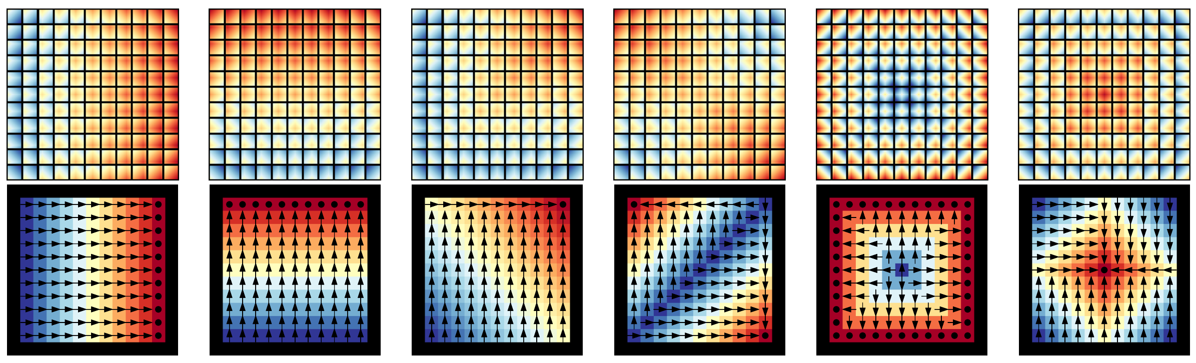
□

Having established a De Morgan algebra over tasks and extended value functions, we show that there exists an equivalence between the two. As a result, if we can write down a task under the De Morgan algebra, we can immediately write down the optimal value function for the task.

Theorem 5.2.2. *Let $\mathcal{M}$ be a set of deterministic shortest path tasks or a set of discounted goal-reaching tasks satisfying Assumption 3.4.1. Let $\bar{\mathcal{Q}}^*$ be the set of optimal $\bar{Q}$-value functions for tasks in $\mathcal{M}$. Let $\mathcal{F} : \mathcal{M} \to \bar{\mathcal{Q}}^*$ be any map from $\mathcal{M}$ to $\bar{\mathcal{Q}}^*$ such that $\mathcal{F}(M) = \bar{Q}^*_M$ for all M in $\mathcal{M}$. Then $\mathcal{F}$ is a homomorphism between the De Morgan task algebra $(\mathcal{M}, \vee, \wedge, \neg, \mathcal{M}_{SUP}, \mathcal{M}_{INF})$ and the De Morgan EVFs algebra $(\bar{\mathcal{Q}}^*, \vee, \wedge, \neg, \bar{Q}^*_{SUP}, \bar{Q}^*_{INF})$.*

Proof. Follows from the proof of Proposition 5.2.4 and the homomorphism between the task and EVF lattices. □

Experiment 5.2.2. *Consider the simple grid world domain introduced in Example 3.4.1 where an agent may be asked to go to any position on the xy-plane. We use the EVFs learned in Experiment 5.2.1 for the tasks M_L and M_D. Figures 5.3 and 5.4 shows the values of the EVFs specified by their negation, $\neg\bar{Q}^*_L = \bar{Q}^*_R$ and $\neg\bar{Q}^*_D = \bar{Q}^*_T$, in which an agent determines how to navigate to the right and top of the xy-plane respectively. This shows that the negation defined above does indeed have the expected semantics. The figure also shows that arbitrary disjunction, conjunction, and negation of EVFs also produces EVFs with the desired semantics.*

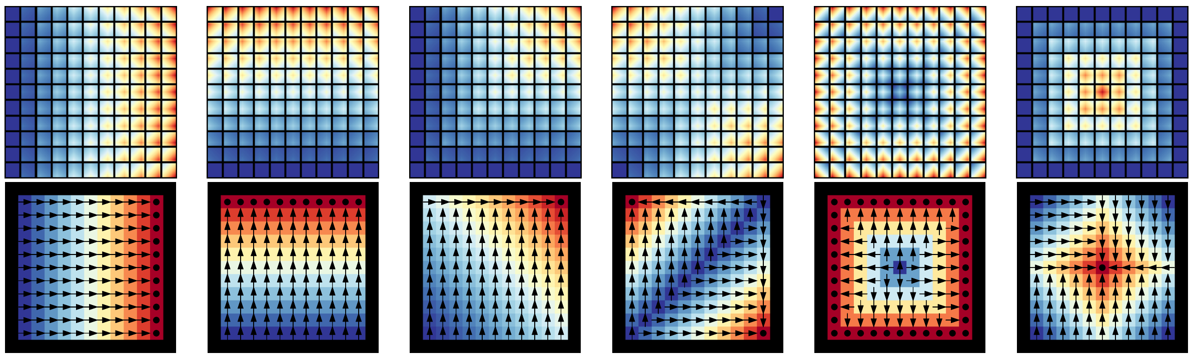


(a) $\neg\bar{Q}^*_{\mathrm{L}}$ (b) $\neg\bar{Q}^*_{\mathrm{D}}$ (c) $\bar{Q}^*_{\mathrm{L}} \bar{\vee} \bar{Q}^*_{\mathrm{D}}$ (d) $\bar{Q}^*_{\mathrm{L}} \underline{\vee} \bar{Q}^*_{\mathrm{D}}$ (e) $(\bar{Q}^*_{\mathrm{L}} \vee \neg\bar{Q}^*_{\mathrm{L}}) \vee (\bar{Q}^*_{\mathrm{D}} \vee \neg\bar{Q}^*_{\mathrm{D}})$ (f) $(\bar{Q}^*_{\mathrm{L}} \wedge \neg\bar{Q}^*_{\mathrm{L}}) \wedge (\bar{Q}^*_{\mathrm{D}} \wedge \neg\bar{Q}^*_{\mathrm{D}})$

Figure 5.3: Showing the composition of EVFs learned from deterministic shortest path tasks in gridworld domain. Figures (a)-(b) shows the values for the right and up EVFs, specified by the negation of the left and down EVFs respectively. (c)-(f) shows the values of the composed EVFs.

5.2.3 Boolean EVF Algebra

While the De Morgan EVF algebra allows us to specify arbitrary disjunction, conjunction, and negation of EVFs, it does not in general represent the full desired properties of logics. In particular the EVF compositions does not always satisfy the laws of the excluded middle and of non contradiction. This is because the De Morgan EVF algebra allows for



(a) $\neg\bar{Q}^*_{\text{L}}$ (b) $\neg\bar{Q}^*_{\text{D}}$ (c) $\bar{Q}^*_{\text{L}} \barvee \bar{Q}^*_{\text{D}}$ (d) $\bar{Q}^*_{\text{L}} \veebar \bar{Q}^*_{\text{D}}$ (e) $\begin{matrix}(\bar{Q}^*_{\text{L}} \vee \neg\bar{Q}^*_{\text{L}}) \\ \vee \\ (\bar{Q}^*_{\text{D}} \vee \neg\bar{Q}^*_{\text{D}})\end{matrix}$ (f) $\begin{matrix}(\bar{Q}^*_{\text{L}} \wedge \neg\bar{Q}^*_{\text{L}}) \\ \wedge \\ (\bar{Q}^*_{\text{D}} \wedge \neg\bar{Q}^*_{\text{D}})\end{matrix}$

Figure 5.4: Showing the composition of EVFs learned from discounted tasks in gridworld domain. Figures (a)-(b) shows the values for the right and up EVFs, specified by the negation of the left and down EVFs respectively. (c)-(f) shows the values of the composed EVFs.

EVFs obtained from tasks with dense rewards in the goal space which in turn violates these assumptions in general. In this section we show that by restricting the EVFs to those of Boolean tasks, we obtain the full logics on EVFs.

Proposition 5.2.6. *Let $\mathcal{M}$ be a set of deterministic shortest path tasks or a set of discounted goal-reaching tasks satisfying Assumption 3.4.2. Let $\bar{\mathcal{Q}}^*$ be the set of optimal $\bar{Q}$-value functions for tasks in $\mathcal{M}$. Then $(\bar{\mathcal{Q}}^*, \vee, \wedge, \neg, \bar{Q}^*_{SUP}, \bar{Q}^*_{INF})$ is a Boolean EVF algebra.*

Proof. Let $\bar{Q}^*_{M_1}, \bar{Q}^*_{M_2} \in \bar{\mathcal{Q}}^*$ be the optimal $\bar{Q}$-value functions for tasks $M_1, M_2 \in \mathcal{M}$ with reward functions r_{M_1} and r_{M_2}. We show that $\neg, \vee, \wedge$ satisfy the Boolean properties (i) – (vii).

(i)–(vi): These follow from the De Morgan EVF algebra since $\bar{\mathcal{Q}}^*$ satisfies its assumptions.

(vii): For all (s, g, a) in $\mathcal{S} \times \mathcal{G} \times \mathcal{A}$,

if $\mathcal{M}$ is a set of deterministic shortest path tasks, then

$$\begin{aligned}
&(\bar{Q}^*_{M_1} \wedge \neg\bar{Q}^*_{M_1})(s, g, a) \\
&= \inf\{\bar{Q}^*_{M_1}(s, g, a), (\bar{Q}^*_{SUP}(s, g, a) - \bar{Q}^*_{INF}(s, g, a)) - \bar{Q}^*_{M_1}(s, g, a)\} \\
&= G^*_{s:g,a} + \inf\{\bar{r}_{M_1}(s', g, a'), (\bar{r}_{\mathcal{M}_{SUP}}(s', g, a') + \bar{r}_{\mathcal{M}_{INF}}(s', g, a')) - \bar{r}_{M_1}(s', g, a')\} \\
&= G^*_{s:g,a} + \bar{r}_{\mathcal{M}_{INF}}(s', g, a') \\
&= \bar{Q}^*_{INF}(s, g, a).
\end{aligned}$$

If $\mathcal{M}$ is a set of discounted goal-reaching tasks, then

$$\begin{aligned}
&(\bar{Q}^*_{M_1} \wedge \neg\bar{Q}^*_{M_1})(s, g, a) \\
&= \inf\{\bar{Q}^*_{M_1}(s, g, a), (\bar{Q}^*_{SUP}(s, g, a) - \bar{Q}^*_{INF}(s, g, a)) - \bar{Q}^*_{M_1}(s, g, a)\} \\
&= \gamma^{T^*_{s:g,a}} \inf\{r_{M_1}(g, a'), (r_{\mathcal{M}_{SUP}}(g, a') + r_{\mathcal{M}_{INF}}(g, a')) - r_{M_1}(g, a')\} \\
&= \gamma^{T^*_{s:g,a}} r_{\mathcal{M}_{INF}}(g, a') \\
&= \bar{Q}^*_{INF}(s, g, a).
\end{aligned}$$

Similarly, $\bar{Q}^*_{M_1} \vee \neg\bar{Q}^*_{M_1} = \bar{Q}^*_{SUP}$.

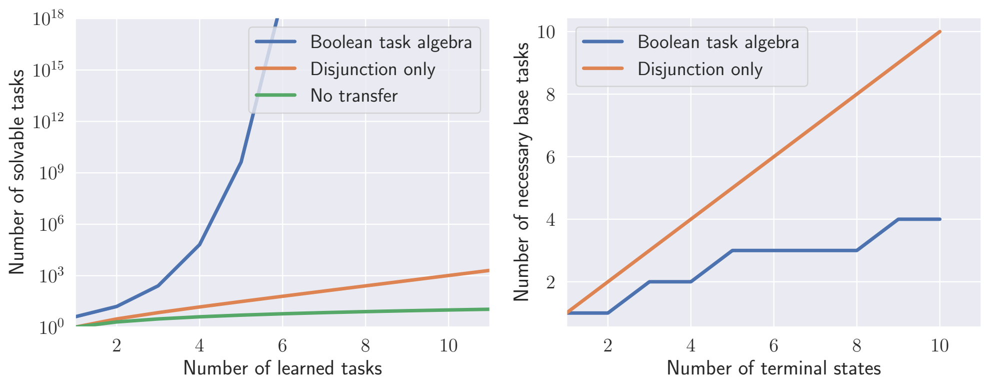
□

Having established a Boolean algebra over tasks and extended value functions, we show that they are homomorphic. As a result, if we can write down a task under the Boolean algebra, we can immediately write down the optimal value function for the task.

Theorem 5.2.3. *Let $\mathcal{M}$ be a set of deterministic shortest path tasks or a set of discounted goal-reaching tasks satisfying Assumption 3.4.2. Let $\bar{\mathcal{Q}}^*$ be the set of optimal $\bar{Q}$-value functions for tasks in $\mathcal{M}$. Let $\mathcal{F} : \mathcal{M} \to \bar{\mathcal{Q}}^*$ be any map from $\mathcal{M}$ to $\bar{\mathcal{Q}}^*$ such that $\mathcal{F}(M) = \bar{Q}^*_M$ for all M in $\mathcal{M}$. Then $\mathcal{F}$ is a homomorphism between the Boolean task algebra $(\mathcal{M}, \vee, \wedge, \neg, \mathcal{M}_{SUP}, \mathcal{M}_{INF})$ and the Boolean EVFs algebra $(\bar{\mathcal{Q}}^*, \vee, \wedge, \neg, \bar{Q}^*_{SUP}, \bar{Q}^*_{INF})$.*

Proof. Follows from the homomorphism between the De Morgan task and EVF algebras. □

Combining Theorem 5.2.3 with the notion of base tasks and super-exponential task specifications obtained in Section 3.5, we obtain the much desired super-exponential explosion of skills (Figure 5.5).



(a) Number of tasks that can be solved as a function of the number of existing tasks solved. Results are plotted on a log-scale.

(b) Number of base tasks that need to be solved to span all tasks as a function of number of terminal states. Results are plotted on a log-scale.

Figure 5.5: Results in comparison to the disjunctive composition of Van Niekerk *et al.* (2019). (a) The extended value functions allow us to solve exponentially more tasks than the disjunctive approach without further learning. (b) With extended value functions the number of base tasks required to solve all tasks is logarithmic in the number of achievable goals.

Experiment 5.2.3. *Consider the simple grid world domain introduced in Example 3.4.3 where an agent may be asked to go to any position on the xy-plane and the goal rewards are restricted to r_{MIN} or r_{MAX}. The agent learns tasks specified as deterministic shortest paths and discounted goal reaching, where*

- *Deterministic shortest path tasks: The internal rewards are -0.1 and the goal rewards are -0.1 for undesired terminal states and 2 for desired ones.*
- *Discounted goal-reaching tasks: The internal rewards are 0 and the goal rewards are 0 for undesired terminal states and 2 for desired ones. The discounting used is $\gamma = 0.95$.*

We train an agent on the tasks M_L and M_D, giving us their respective EVFs $\bar{Q}^*_L$ and $\bar{Q}^*_D$. We are now able to do the zero-shot composition of any logical combination of M_L and M_D. This is demonstrated in Figures 5.6 and 5.7. As usual we observe that the EVFs have the same structure as the rewards in the absorbing set, with the EVFs of deterministic shortest path tasks learning how to achieve any absorbing state while that of discounted goal-reaching tasks learn to achieve nothing for absorbing states with zero reward. Notice how for the lower bound EVF (the result of meaningless composition), the optimal policy for the deterministic shortest path is to achieve any terminal state (Figure 5.6(f)). This is because all the terminal states are equally undesirable (they have the lowest values). In contrast we observe that the EVF for meaningless compositions of discounted goal-reaching tasks is zero everywhere since the agent is never rewarded for any action in any state (Figure 5.7(f)). Hence the agent learns nothing (good or bad), and as a result an optimal policy here is to achieve nothing.

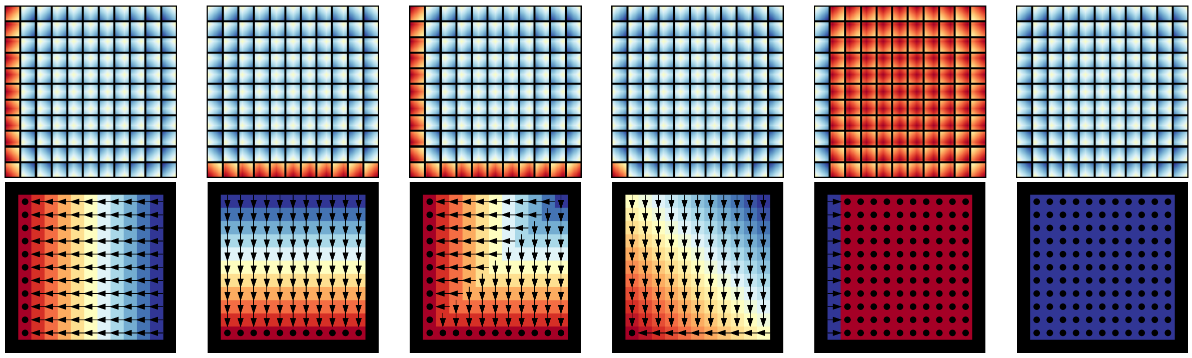

(a) $\bar{Q}^*_{\mathrm{L}}$ (b) $\bar{Q}^*_{\mathrm{D}}$ (c) $\bar{Q}^*_{\mathrm{L}} \vee \bar{Q}^*_{\mathrm{D}}$ (d) $\bar{Q}^*_{\mathrm{L}} \wedge \bar{Q}^*_{\mathrm{D}}$ (e) $\neg\bar{Q}^*_{\mathrm{L}}$ (f) $\begin{matrix}(\bar{Q}^*_{\mathrm{L}} \wedge \neg\bar{Q}^*_{\mathrm{L}}) \\ \wedge \\ (\bar{Q}^*_{\mathrm{D}} \wedge \neg\bar{Q}^*_{\mathrm{D}})\end{matrix}$

Figure 5.6: Showing composition of Boolean EVFs in gridworld domain. Figures (a)-(b) shows the Boolean goal rewards for the left and down tasks, (c)-(f) shows the rewards of the composed tasks.

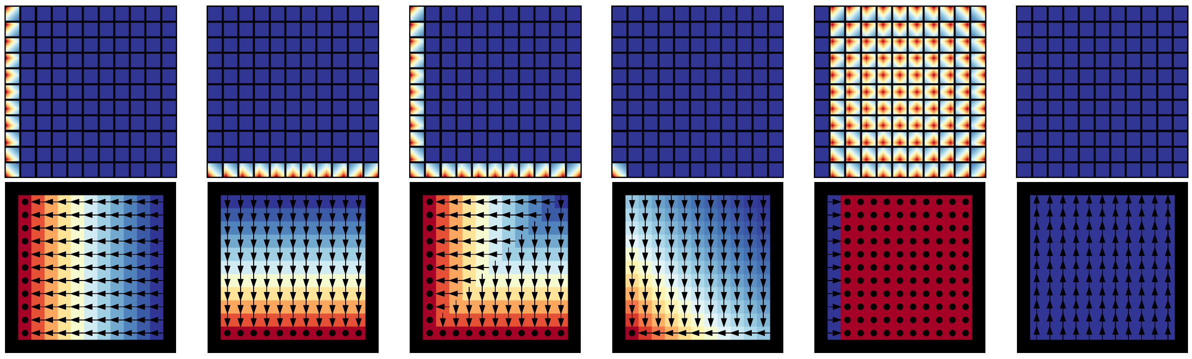

(a) $\bar{Q}^*_{\mathrm{L}}$ (b) $\bar{Q}^*_{\mathrm{D}}$ (c) $\bar{Q}^*_{\mathrm{L}} \vee \bar{Q}^*_{\mathrm{D}}$ (d) $\bar{Q}^*_{\mathrm{L}} \wedge \bar{Q}^*_{\mathrm{D}}$ (e) $\neg\bar{Q}^*_{\mathrm{L}}$ (f) $\begin{matrix}(\bar{Q}^*_{\mathrm{L}} \wedge \neg\bar{Q}^*_{\mathrm{L}}) \\ \wedge \\ (\bar{Q}^*_{\mathrm{D}} \wedge \neg\bar{Q}^*_{\mathrm{D}})\end{matrix}$

Figure 5.7: Showing composition of Boolean tasks in gridworld domain. Figures (a)-(b) shows the Boolean goal rewards for the left and down tasks, (c)-(f) shows the rewards of the composed tasks.

5.3 Between EVF and Power Set Boolean Algebras

In the previous section we showed that the algebraic structures of tasks and EVFs are homomorphic. As a result any task that is specified according to a task algebra can be immediately solved according to the EVF algebra. In this section we show that for the case of a Boolean EVF algebra, we can in fact immediately construct optimal EVFs directly from a set of desired goals. We do this by showing that the EVF and power set Boolean algebras are in fact isomorphic. Consider the mapping F between the set of EVFs $\bar{\mathcal{Q}}^*$ and the power-set $\mathcal{P}(\mathcal{G})$, given by

$$\begin{aligned} F:\ & \mathcal{P}(\mathcal{G}) \to \bar{\mathcal{Q}}^* \\ & H \mapsto \bar{Q}^*_H, \text{ where } \bar{Q}^*_H:\ \mathcal{S}\times\mathcal{G}\times\mathcal{A} \to \mathbb{R} \\ & \qquad (s,g,a) \mapsto \begin{cases} \bar{Q}^*_{SUP}(s,g,a), & \text{if } g \in H \\ \bar{Q}^*_{INF}(s,g,a), & \text{otherwise.} \end{cases} \end{aligned}$$

F is clearly an isomorphism since each $g \in H$ defines and can be defined by each $g' \in \mathcal{G}$ that gives $\bar{Q}^*_H(s,g',a) = \bar{Q}^*_{SUP}(s,g',a)\ \forall\ (s,a) \in \mathcal{S}\times\mathcal{A}$. The isomorphism between a Boolean EVF algebra and the power-set Boolean algebra gives us the following important result.

Theorem 5.3.1. *Let $\mathcal{M}$ be a set of deterministic shortest path tasks or a set of discounted goal-reaching tasks satisfying Assumption 3.4.2. Let $\bar{\mathcal{Q}}^*$ be the set of optimal $\bar{Q}$-value functions for tasks in $\mathcal{M}$. Then the Boolean algebra on $\bar{\mathcal{Q}}^*$ is isomorphic to the Boolean algebra on $\mathcal{M}$.*

Proof. This follows from the isomorphism between $\bar{\mathcal{Q}}^*$ and $\mathcal{P}(\mathcal{G})$, and that between $\mathcal{P}(\mathcal{G})$ and $\mathcal{M}$. □

This illustrates how the base knowledge an agent needs to act optimally in an environment for any future task can in fact be constructed rather than learned. All that is required to be learned are two EVFs, namely the lower bound EVF $\bar{Q}^*_{INF}$ and upper bound EVF $\bar{Q}^*_{SUP}$. Note how this is true for any cardinality of $\mathcal{G}$. If $\mathcal{G}$ is finite then this can easily be done with a Boolean table.

Experiment 5.3.1. *Consider a Four Rooms domain described in Section 3.5, where an agent must navigate in a grid world to particular rooms. The agent learns tasks specified as deterministic shortest paths and discounted goal reaching. The rewards for both cases are the same as in Experiment 5.2.3.*

Just as in the Boolean task algebra, we can select a minimal set of base EVFs (generator EVFs) by assigning each goal a binary number, and then using the columns of the table to generate the EVFs. Since the absorbing set is finite, we can do this assignment using a Boolean table. Similarly to Section 3.5, we label each goal by defining a well order over $\mathcal{G}$. The number of base EVFs induced are then $\lceil|\log_2 \mathcal{G}|\rceil = 2$ since $|\mathcal{G}| = 4$. Table 5.1 illustrates how different well orders on $\mathcal{G}$ leads to different choices of base EVFs.

Consider the well order on $\mathcal{G}$ given by Table 3.2(b). The base EVFs induced are $\bar{Q}^*_L$ and $\bar{Q}^*_T$ which are the EVFs of the tasks M_L and M_T respectively.

Figures 5.8 and 5.9 illustrates these EVFs for the two classes of MDPs and some of their logical compositions. Note that the resulting optimal value function can attain a goal not explicitly represented by the base tasks, namely the bottom-right goal for the negation. Figures 5.10 and 5.11 shows the Boolean table and Hasse diagram for all the 16 EVFs generated by them, which spans their respective value spaces.

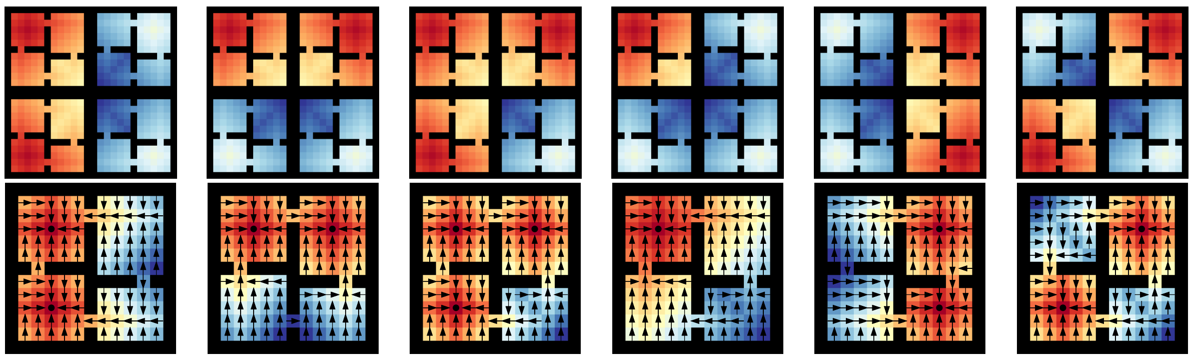
Goals	$\bar{Q}^*_D$	$\bar{Q}^*_R$
top-left	0	0
top-right	0	1
bottom-left	1	0
bottom-right	1	1

(a) Goals labled by the well order $\leq$ given by: $bottom-right \leq bottom-left \leq top-right \leq top-left$

Goals	$\bar{Q}^*_T$	$\bar{Q}^*_L$
bottom-right	0	0
bottom-left	0	1
top-right	1	0
top-left	1	1

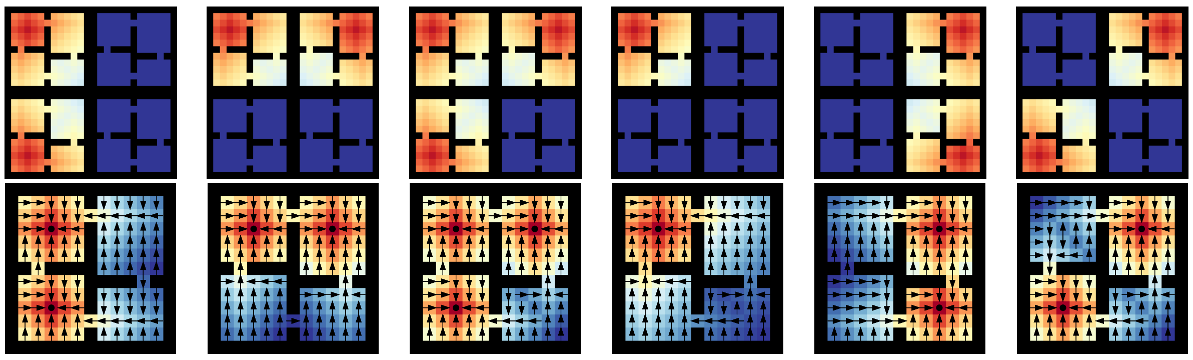
(b) Goals labled by the well order $\leq$ given by: $top-left \leq top-right \leq bottom-left \leq bottom-right$

Table 5.1: Base EVFs induced by various well orders on $\mathcal{G}$. Each column represents a base EVF, where **0** or **1** for goal g on EVF $\bar{Q}^*$ means respectively EVF of $\bar{Q}^*(s,g,a) = \bar{Q}^*_{SUP}(s,g,a)$ or $\bar{Q}^*(s,g,a) = \bar{Q}^*_{INF}(s,g,a)$ $\forall(s,a) \in \mathcal{S} \times \mathcal{A}$.



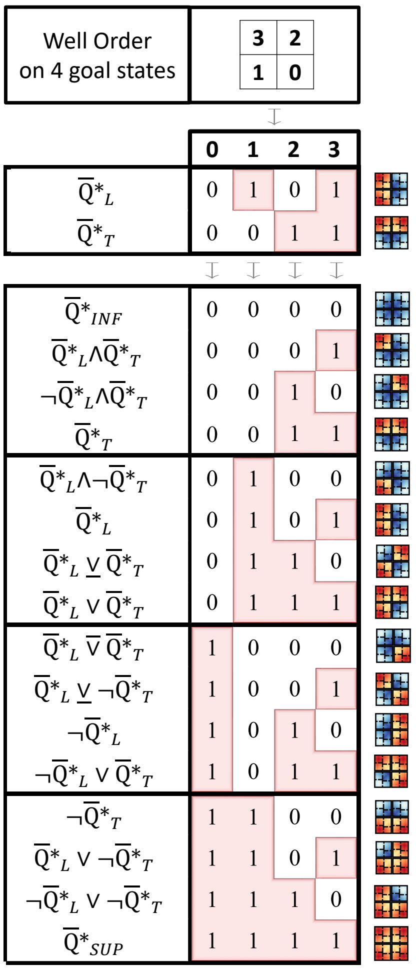
(a) $\bar{Q}^*_L$ (b) $\bar{Q}^*_T$ (c) $\bar{Q}^*_L \vee \bar{Q}^*_T$ (d) $\bar{Q}^*_L \wedge \bar{Q}^*_T$ (e) $\neg\bar{Q}^*_L$ (f) $\bar{Q}^*_L \veebar \bar{Q}^*_T$

Figure 5.8: An example of Boolean algebraic composition using generated extended value functions of base deterministic shortest path tasks. Arrows represent the optimal action in a given state. (a–b) The generated optimal extended value functions for the base tasks. (c) Zero-shot disjunctive composition. (d) Zero-shot conjunctive composition. (e) Zero-shot negation. (f) Combining operators to model exclusive-or composition.



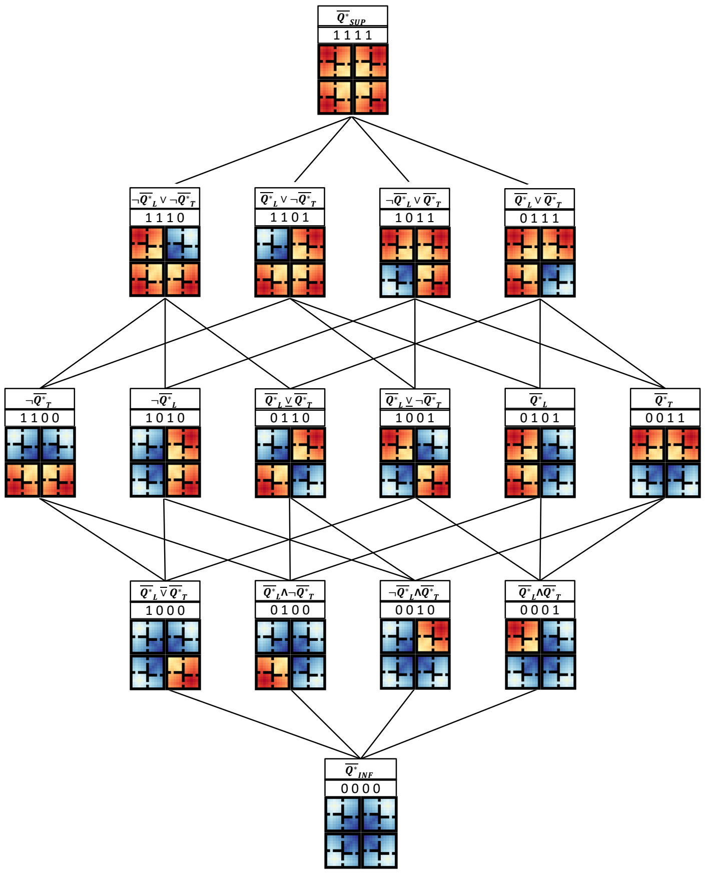
(a) $\bar{Q}^*_L$ (b) $\bar{Q}^*_T$ (c) $\bar{Q}^*_L \vee \bar{Q}^*_T$ (d) $\bar{Q}^*_L \wedge \bar{Q}^*_T$ (e) $\neg\bar{Q}^*_L$ (f) $\bar{Q}^*_L \veebar \bar{Q}^*_T$

Figure 5.9: An example of Boolean algebraic composition using generated extended value functions of base discounted goal-reaching tasks. Arrows represent the optimal action in a given state. (a–b) The generated optimal extended value functions for the base tasks. (c) Zero-shot disjunctive composition. (d) Zero-shot conjunctive composition. (e) Zero-shot negation. (f) Combining operators to model exclusive-or composition.



Well Order on 4 goal states	3 2 / 1 0			
	0	1	2	3
$\bar{Q}^*_L$	0	1	0	1
$\bar{Q}^*_T$	0	0	1	1
$\bar{Q}^*_{INF}$	0	0	0	0
$\bar{Q}^*_L \wedge \bar{Q}^*_T$	0	0	0	1
$\neg\bar{Q}^*_L \wedge \bar{Q}^*_T$	0	0	1	0
$\bar{Q}^*_T$	0	0	1	1
$\bar{Q}^*_L \wedge \neg\bar{Q}^*_T$	0	1	0	0
$\bar{Q}^*_L$	0	1	0	1
$\bar{Q}^*_L \veebar \bar{Q}^*_T$	0	1	1	0
$\bar{Q}^*_L \vee \bar{Q}^*_T$	0	1	1	1
$\bar{Q}^*_L \bar{\vee} \bar{Q}^*_T$	1	0	0	0
$\bar{Q}^*_L \veebar \neg\bar{Q}^*_T$	1	0	0	1
$\neg\bar{Q}^*_L$	1	0	1	0
$\neg\bar{Q}^*_L \vee \bar{Q}^*_T$	1	0	1	1
$\neg\bar{Q}^*_T$	1	1	0	0
$\bar{Q}^*_L \vee \neg\bar{Q}^*_T$	1	1	0	1
$\neg\bar{Q}^*_L \vee \neg\bar{Q}^*_T$	1	1	1	0
$\bar{Q}^*_{SUP}$	1	1	1	1

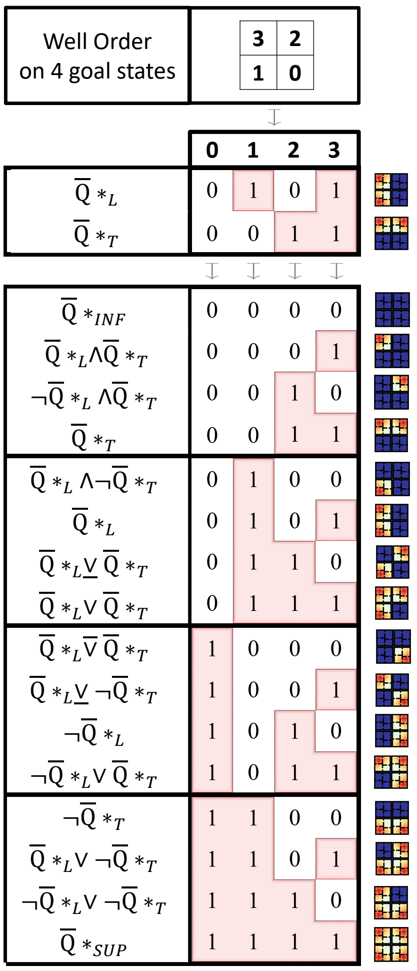
(a) Boolean table of base and composed EVFs.



$\bar{Q}^*_{SUP}$ 1111
$\neg\bar{Q}^*_L \vee \neg\bar{Q}^*_T$ 1110
$\bar{Q}^*_L \vee \neg\bar{Q}^*_T$ 1101
$\neg\bar{Q}^*_L \vee \bar{Q}^*_T$ 1011
$\bar{Q}^*_L \vee \bar{Q}^*_T$ 0111
$\neg\bar{Q}^*_T$ 1100
$\neg\bar{Q}^*_L$ 1010
$\bar{Q}^*_L \veebar \bar{Q}^*_T$ 0110
$\bar{Q}^*_L \veebar \neg\bar{Q}^*_T$ 1001
$\bar{Q}^*_L$ 0101
$\bar{Q}^*_T$ 0011
$\bar{Q}^*_L \bar{\vee} \bar{Q}^*_T$ 1000
$\bar{Q}^*_L \wedge \neg\bar{Q}^*_T$ 0100
$\neg\bar{Q}^*_L \wedge \bar{Q}^*_T$ 0010
$\bar{Q}^*_L \wedge \bar{Q}^*_T$ 0001
$\bar{Q}^*_{INF}$ 0000

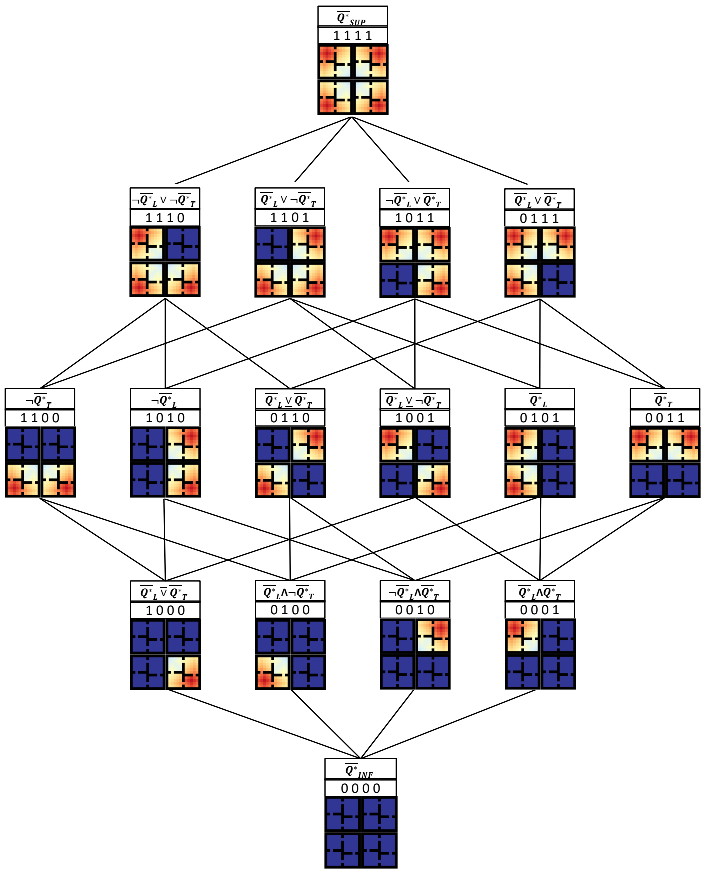
(b) Hasse diagram of the Boolean EVF algebra.

Figure 5.10: Boolean table and Hasse diagramm for EVFs of deterministic shortest path tasks in the the four-rooms domain. (a) shows a well order on $\mathcal{G}$ that labels the goals, the induced base EVFs from labeling the goals, the logical expressions for all 16 compositions of the base EVFs, their Boolean values and value functions per goal. (b) illustrates the Boolean EVF algebra showing the EVFs for all 16 tasks in $\mathcal{M}$, together with the Boolean values and logical expressions that generates them from the base EVFs.



	0	1	2	3
$\bar{Q}*_L$	0	1	0	1
$\bar{Q}*_T$	0	0	1	1
$\bar{Q}*_{INF}$	0	0	0	0
$\bar{Q}*_L \wedge \bar{Q}*_T$	0	0	0	1
$\neg\bar{Q}*_L \wedge \bar{Q}*_T$	0	0	1	0
$\bar{Q}*_T$	0	0	1	1
$\bar{Q}*_L \wedge \neg\bar{Q}*_T$	0	1	0	0
$\bar{Q}*_L$	0	1	0	1
$\bar{Q}*_L \underline{\vee} \bar{Q}*_T$	0	1	1	0
$\bar{Q}*_L \vee \bar{Q}*_T$	0	1	1	1
$\bar{Q}*_L \bar{\vee} \bar{Q}*_T$	1	0	0	0
$\bar{Q}*_L \underline{\vee} \neg\bar{Q}*_T$	1	0	0	1
$\neg\bar{Q}*_L$	1	0	1	0
$\neg\bar{Q}*_L \vee \bar{Q}*_T$	1	0	1	1
$\neg\bar{Q}*_T$	1	1	0	0
$\bar{Q}*_L \vee \neg\bar{Q}*_T$	1	1	0	1
$\neg\bar{Q}*_L \vee \neg\bar{Q}*_T$	1	1	1	0
$\bar{Q}*_{SUP}$	1	1	1	1

(a) Boolean table of base and composed EVFs.


$\bar{Q}^*_{SUP}$ 1111
$\neg\bar{Q}^*_L \vee \neg\bar{Q}^*_T$ 1110
$\bar{Q}^*_L \vee \neg\bar{Q}^*_T$ 1101
$\neg\bar{Q}^*_L \vee \bar{Q}^*_T$ 1011
$\bar{Q}^*_L \vee \bar{Q}^*_T$ 0111
$\neg\bar{Q}^*_T$ 1100
$\neg\bar{Q}^*_L$ 1010
$\bar{Q}^*_L \underline{\vee} \bar{Q}^*_T$ 0110
$\bar{Q}^*_L \underline{\vee} \neg\bar{Q}^*_T$ 1001
$\bar{Q}^*_L$ 0101
$\bar{Q}^*_T$ 0011
$\bar{Q}^*_L \bar{\vee} \bar{Q}^*_T$ 1000
$\bar{Q}^*_L \wedge \neg\bar{Q}^*_T$ 0100
$\neg\bar{Q}^*_L \wedge \bar{Q}^*_T$ 0010
$\bar{Q}^*_L \wedge \bar{Q}^*_T$ 0001
$\bar{Q}^*_{INF}$ 0000

(b) Hasse diagram of the Boolean EVF algebra.

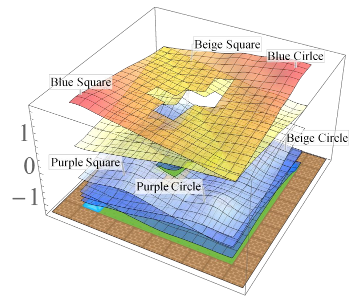
Figure 5.11: Boolean table and Hasse diagram for EVFs of discounted goal-reaching tasks in the four-rooms domain. (a) shows a well order on $\mathcal{G}$ that labels the goals, the induced base EVFs from labeling the goals, the logical expressions for all 16 compositions of the base EVFs, their Boolean values and value functions per goal. (b) illustrates the Boolean EVF algebra showing the EVFs for all 16 tasks in $\mathcal{M}$, together with the Boolean values and logical expressions that generates them from the base EVFs.

5.4 Zero-Shot Composition With Function Approximation

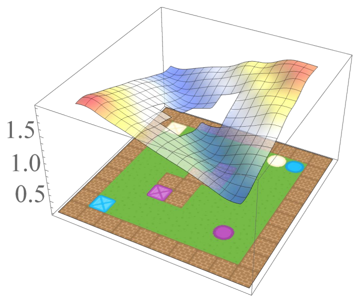
Finally, we demonstrate that our compositional approach can also be used to tackle high-dimensional domains where function approximation is required. We use the same video game environment as in Section 4.4, where the observations are images of the 2D game world and the agent must navigate to collect objects of different shapes and colours. We show zero-shot composition when the tasks are specified as deterministic shortest paths and discounted goal reaching.

5.4.1 Deterministic Shortest Path Tasks

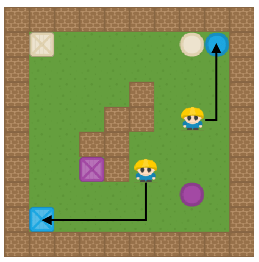
Similarly to Section 4.4, for each task the agent receives a reward of 2 when it collects desired objects and −0.1 everywhere else. We first learn to solve two base tasks: collecting blue objects, and collecting squares, which can then be composed to solve new tasks immediately. Figure 5.12 shows the learned extended value functions, their value functions obtained by maximizing over goals, and sample trajectories.

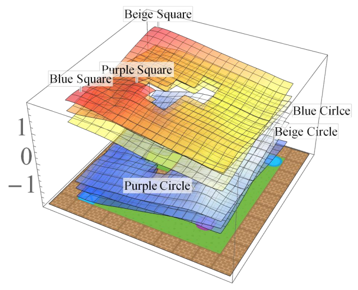


(a) Extended value functions for collecting blue objects.

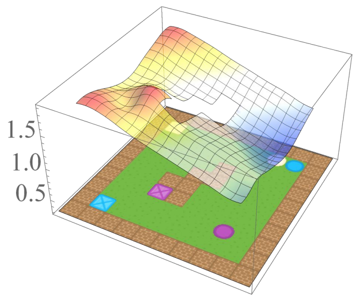


(b) Value functions for collecting blue objects.


(c) Trajectories for collecting blue objects.



(d) Extended value function for collecting squares.



(e) Value function for collecting squares.

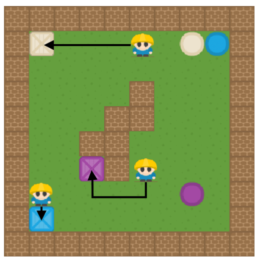
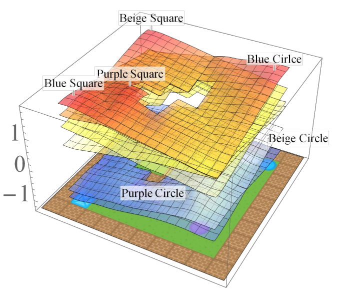
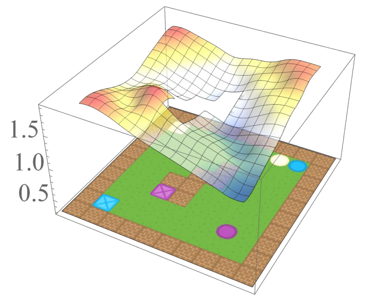

(f) Trajectories for collecting squares.

Figure 5.12: Base tasks for the video game environment in the deterministic shortest path setting. We show extended value functions learned for the blue, and square tasks.

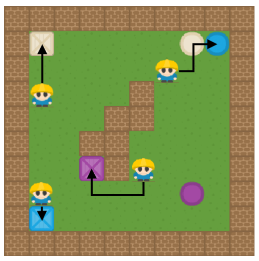
We demonstrate composition characterised by (i) disjunction, (ii) conjunction and (iii) exclusive-or. This corresponds to tasks where the target items are: (i) blue or square, (ii) blue squares, and (iii) blue or squares, but not blue squares. Figure 5.13 illustrates extended value functions, as well as value functions obtained by maximizing over goals and sample trajectories for the respective tasks. Notice how even with function approximation composition of EVFs still produces EVFs. That is the composed EVFs not only lead to optimal value functions and trajectories, but they still retain the properties of EVFs like optimal values for each goal. This is why in general, EVFs allow for as many compositions as desired.

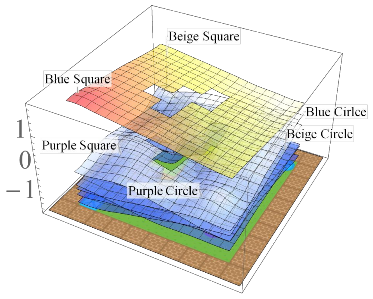


(a) Extended value function for disjunctive composition.

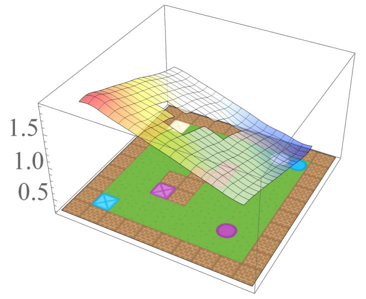


(b) Value function for disjunctive composition.

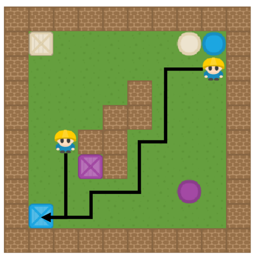

(c) Trajectory for disjunctive composition.

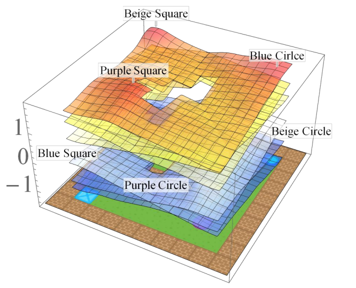


(d) Extended value function for conjunctive composition.

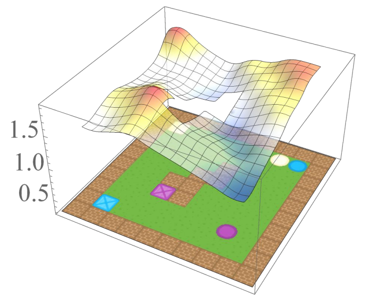


(e) Value function for conjunctive composition.


(f) Trajectories for conjunctive composition.



(g) Extended value function for exclusive-or composition.



(h) Value function for exclusive-or composition.

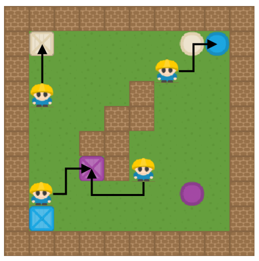
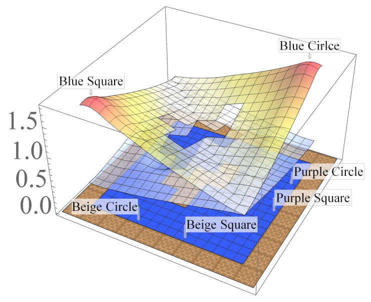

(i) Trajectories for exclusive-or composition.

Figure 5.13: By composing extended value functions from the base deterministic shortest path tasks (collecting blue objects, and collecting squares), we can act optimally in new tasks with no further learning. The extended value functions shows the value function with respect to each goal plotted on the same axis. To generate this, we place the agent at every location and apply the element wise composition operators on the outputs of each network. We then compute the maximum over all actions for each goal. The points are interpolate between each other to smooth the graph. Any error in the visualisation is due to the use of non-linear function approximation.

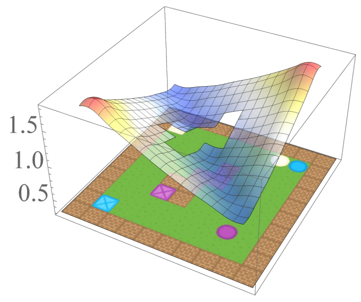
5.4.2 Discounted goal-reaching tasks

Similarly to Section 4.4, for each discounted goal reaching task the agent receives a reward of 2 when it collects desired objects and 0 everywhere else. The discount factor used is also $\gamma = 0.9$. We first learn to solve two base tasks: collecting blue objects, and collecting squares, which can then be composed to solve new tasks immediately. Figure 5.12 shows the learned extended value functions, their value functions obtained by maximizing over goals, and sample trajectories.

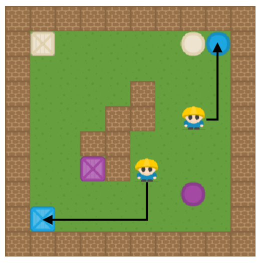
We demonstrate the compositions characterised by (i) disjunction, (ii) conjunction and (iii) exclusive-or. Figure 5.13 illustrates extended value functions, as well as value functions obtained by maximizing over goals and sample trajectories for the respective tasks. We again observe how even with function approximation composition of EVFs still produces EVFs. However unlike deterministic shortest path tasks we note how the EVFs here only learn to collect desired objects. This suggests that even in the case of different terminal states, that is when termination only occurs when collecting desired objects, the EVF of discounted goal-reaching tasks should remain the same.



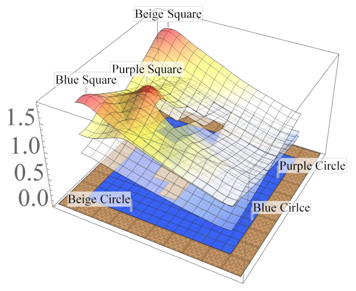
(a) Extended value function for collecting blue objects.



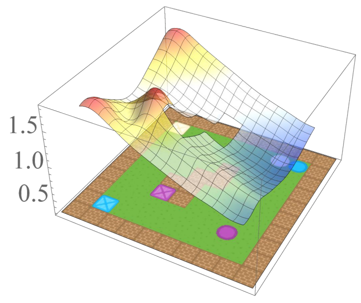
(b) Value function for collecting blue objects.



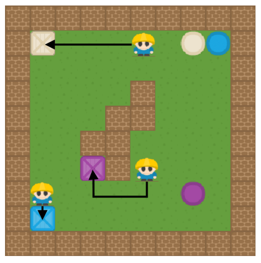
(c) Trajectories for collecting blue objects.



(d) Extended value function for collecting squares.

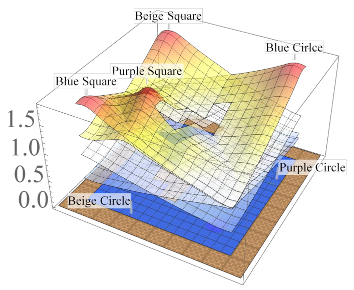


(e) Value function for collecting squares.

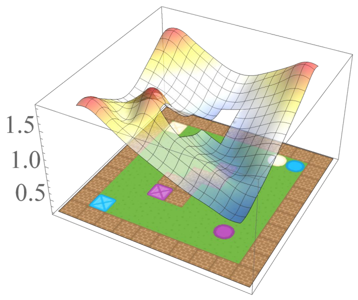


(f) Trajectories for collecting squares.

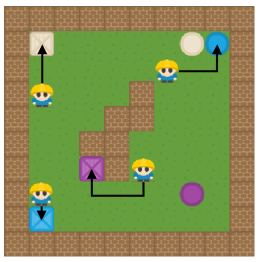
Figure 5.14: Base tasks for the video game environment in the discounted goal reaching setting. We show extended value functions learned for the blue, and square tasks.



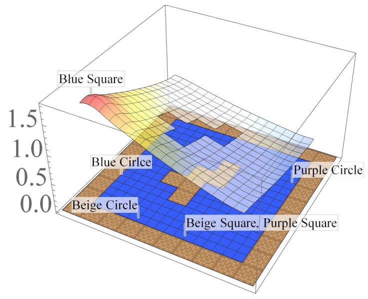
(a) Extended value function for disjunctive composition.



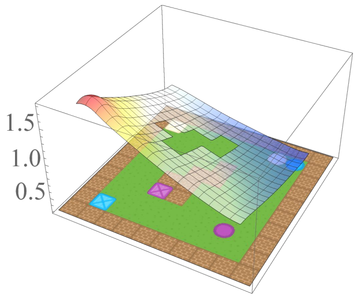
(b) Value function for disjunctive composition.



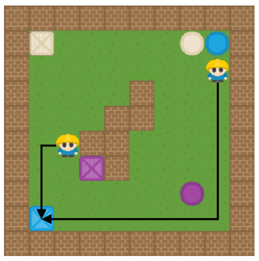
(c) Trajectory for disjunctive composition.



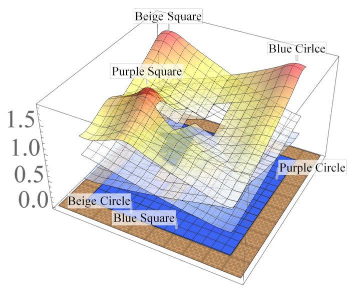
(d) Extended value function for conjunctive composition.



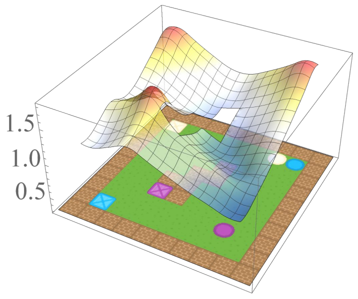
(e) Value function for conjunctive composition.



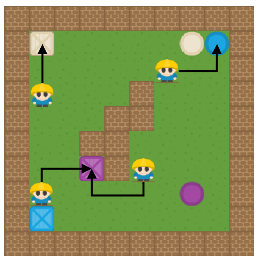
(f) Trajectories for conjunctive composition.



(g) Extended value function for exclusive-or composition.



(h) Value function for exclusive-or composition.



(i) Trajectories for exclusive-or composition.

Figure 5.15: By composing extended value functions from the base discounted goal-reaching tasks (collecting blue objects, and collecting squares), we can act optimally in new tasks with no further learning. The extended value functions shows the value function with respect to each goal plotted on the same axis. To generate this, we place the agent at every location and apply the element wise composition operators on the outputs of each network. We then compute the maximum over all actions for each goal. The points are interpolate between each other to smooth the graph. Any error in the visualisation is due to the use of non-linear function approximation.

5.5 Investigating Practical Considerations

The theoretical results presented in this work rely on Assumptions 3.3.1 and 3.4.2, which restrict the tasks' transition dynamics and reward functions in potentially problematic ways. Although this is necessary to prove that algebraic composition results in optimal value functions, in this section we investigate whether these can be practically ignored. In particular, we investigate two restrictions: the requirement that tasks share the same terminal states, and the impact of using dense rewards on non-terminal states.

5.5.1 Four Rooms Experiments

We use the same setup as the experiment outlined in Section 3.5, but modify it in two ways. We first investigate the difference between using sparse and dense rewards. Our sparse reward function is defined as

$$r_{\text{sparse}}(s, a, s') = \begin{cases} 2 & \text{if } s' \in \mathcal{G} \\ -0.1 & \text{otherwise,} \end{cases}$$

and we use a dense reward function similar to Peng *et al.* (2019):

$$r_{\text{dense}}(s, a, s') = \frac{0.1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} \exp\left(\frac{|s' - g|^2}{4}\right) + r_{\text{sparse}}(s, a, s').$$

Using this dense reward function, we again learn to solve the two base task M_T (reaching the centre of the top two rooms) and M_L (reaching the centre of the left two rooms). We then compose them to solve a variety of tasks, with the resulting value functions illustrated by Figure 5.16.

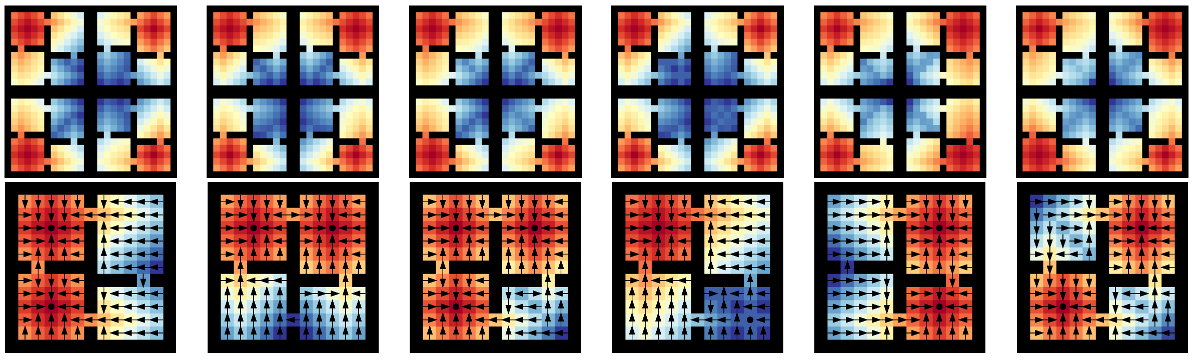

(a) $\bar{Q}^*_L$ (b) $\bar{Q}^*_T$ (c) $\bar{Q}^*_L \vee \bar{Q}^*_T$ (d) $\bar{Q}^*_L \wedge \bar{Q}^*_T$ (e) $\neg \bar{Q}^*_L$ (f) $\bar{Q}^*_L \veebar \bar{Q}^*_T$

Figure 5.16: An example of Boolean algebraic composition using the learned extended value functions with dense rewards. Top row shows extended value functions while bottom one shows resulting value functions and policies obtained by maximizing over goals. Arrows represent the optimal action in a given state. (a–b) The learned optimal extended value functions and resulting value functions for the base tasks with dense rewards. (c) Disjunctive composition. (d) Conjunctive composition. (e) Negation. (f) Combining operators to model exclusive-or composition. We note that the resulting value functions are very similar to those produced in the sparse reward setting.

We also modify the domain so that tasks need not share the same terminating states (that is, if the agent enters a terminating state for a *different* task, the episode does not terminate and the agent can continue as if it were a normal state). This results in four versions of the experiment:

(i) `sparse reward, same absorbing set`

(ii) `sparse reward, different absorbing set`

(iii) `dense reward, same absorbing set`

(iv) `dense reward, different absorbing set`

We learn extended value functions for each of the above setups, and then compose them to solve each of the 2^4 tasks representable in the Boolean algebra. We measure each composed value functions by evaluating its policy in the sparse reward setting, averaging results over 100000 episodes. The results are given by Figure 5.17.

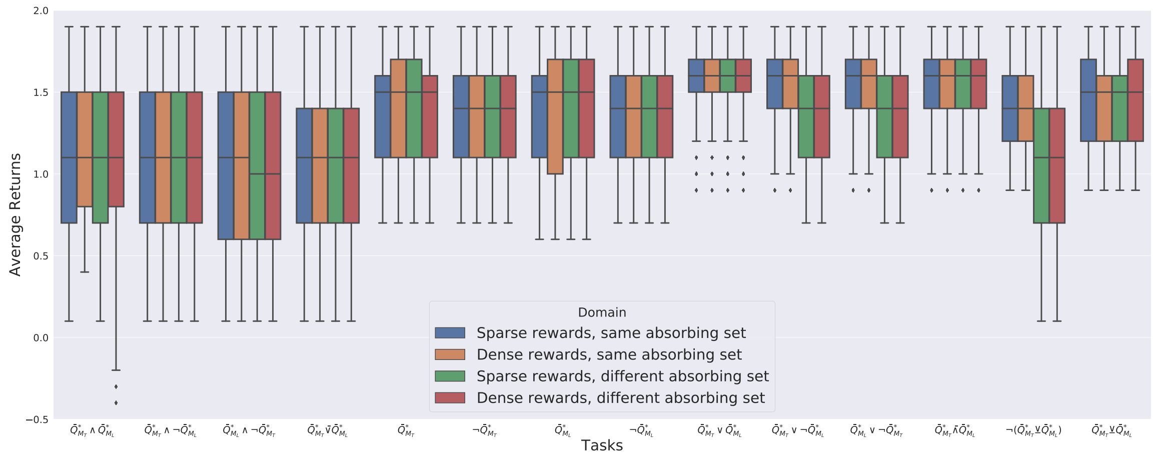


Figure 5.17: Box plots indicating returns for each of the compositional tasks, and for each of the four variations of the domain. Results are collected over 100000 episodes with random start positions.

Our results indicate that extended value functions learned in the theoretically optimal manner (`sparse reward, same absorbing set`) are indeed optimal. However, for the majority of the tasks, relaxing the restrictions on terminal states and reward functions results in policies that are either identical or very close to optimal. Finally, we show that while EVFs are more expensive to learn than regular value functions, the trade-off with the compositional explosion of skills is worth it. We demonstrate this in a modified four-rooms domain with 40 goals, where we need to learn only 7 base tasks, as opposed to 40 for the disjunctive case. Figure 5.18 shows results in comparison to the disjunctive composition of Van Niekerk *et al.* (2019).

5.5.2 Function Approximation Experiments

In this section we investigate whether we can again loosen some of the restrictive assumptions when tackling high-dimensional environments. In particular, we run the same experiments as those presented in Section 5.4, but modify the domain so that (i) tasks need not share the same absorbing set, (ii) the `pickup-up` action is removed (the agent immediately collects an object when reaching it), and (iii) the position of every object is randomised at the start of each episode.

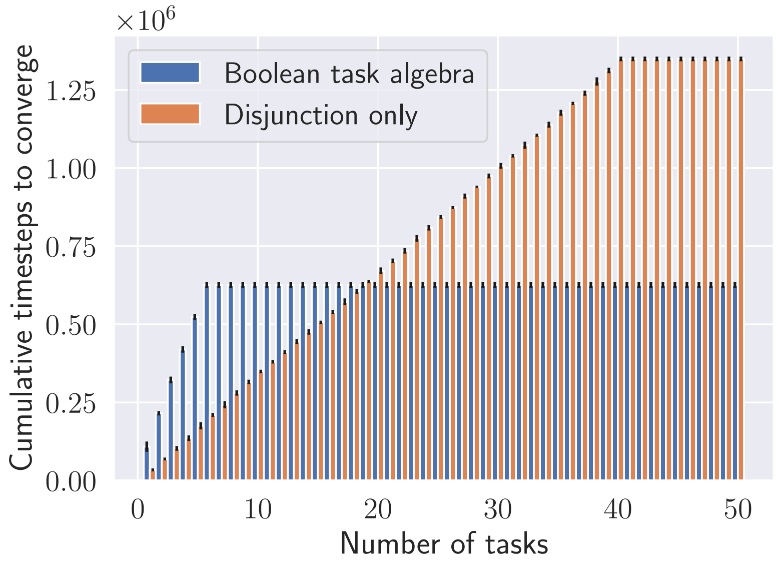


Figure 5.18: Cumulative number of samples required to solve tasks in a 40-goal Four Rooms domain. Error bars represent standard deviations over 100 runs.

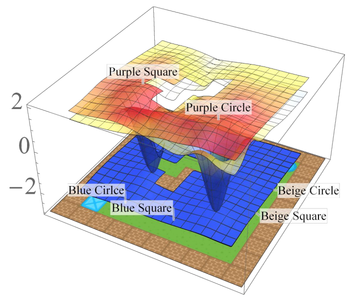
We first learn to solve the three base tasks: collecting blue objects, collecting purple objects, and collecting squares. Their resulting trajectories, value functions and extended value functions are illustrated by Figure 5.19. Notice how even with the relaxed assumptions the EVFs still successfully decouples the values with respect to the desired goals for each task. In fact these EVFs are now similar to that of discounted goal-reaching tasks in that the agent does not learn how to achieve undesired goals, since they are no longer terminating. Note that because the pickup action is removed, the environment terminates upon touching an object and the agent can no longer reach any other object. This results in the large dips in values observed in the shown EVFs.

We then demonstrate that zero-shot compositions do indeed still hold by showing compositions characterised by disjunction, conjunction and exclusive-or, with the resulting trajectories, value functions and extended value functions illustrated by Figure 5.20.

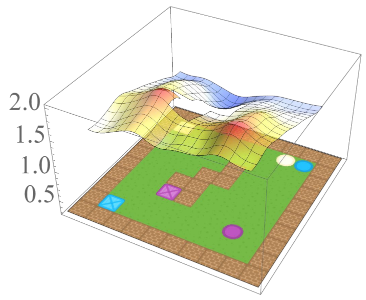
In summary, we have shown that our compositional approach offers strong empirical performance, even when the theoretical assumptions are violated. Finally, we expect that, in general, the errors due to these violations will be far outweighed by the errors due to non-linear function approximation.

5.6 Conclusion

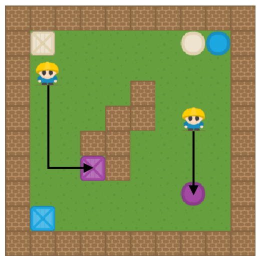
In this chapter we formally showed how EVFs can be composed to solve logical specifications of tasks without extra learning. We demonstrated this zero-shot composition in both gridworld and video game domains. This showed that our composition method does in fact work even in high-dimensional environments whose tasks require function approximation to learn. We demonstrated that in practice our assumptions required for theoretical guarantees can be relaxed while still achieving zero-shot composition. This further solidifies the potential usefulness of this framework towards the goal of having agents in real life that exhibit the much desired compositional explosion of skills.

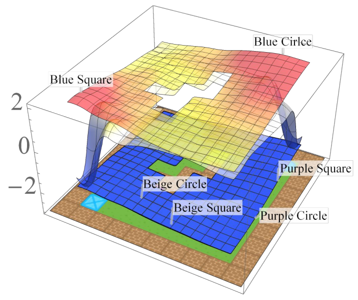


(a) Extended value function for collecting purple objects

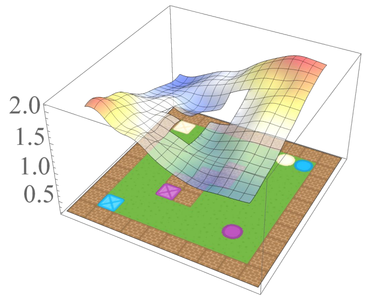


(b) Value function for collecting purple objects

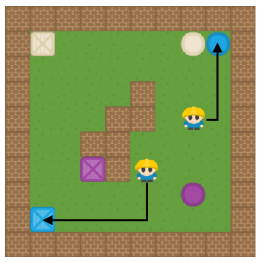

(c) Trajectories for collecting purple objects

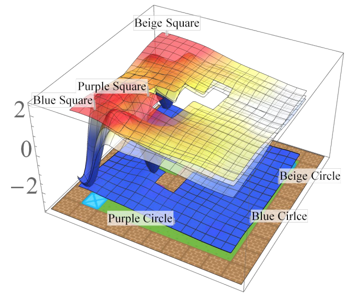


(d) Extended value function for collecting blue objects

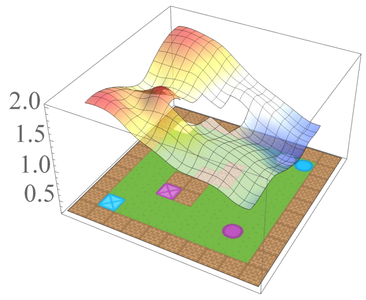


(e) Value function for collecting blue objects


(f) Trajectories for collecting blue objects



(g) Extended value function for collecting squares



(h) Value function for collecting squares

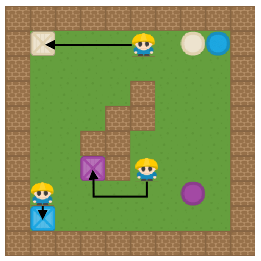
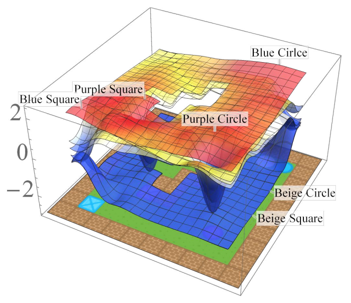
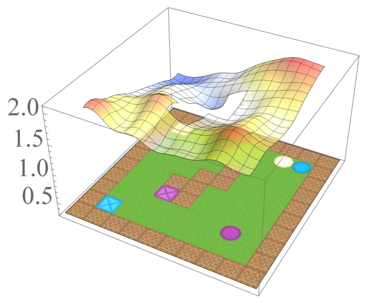

(i) Trajectories for collecting squares

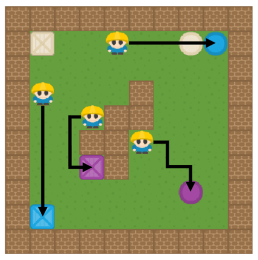
Figure 5.19: Base tasks for the video game environment with relaxed assumptions. We show extended value functions learned for the purple, blue, and square tasks.

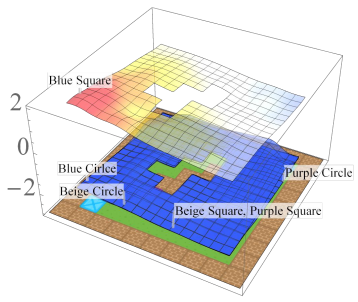


(a) Extended value function for disjunctive composition.

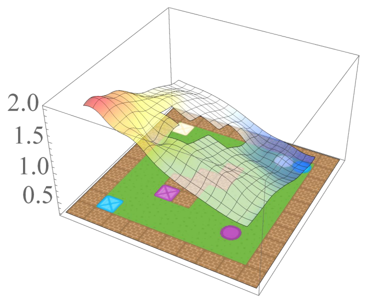


(b) Value function for disjunctive composition.

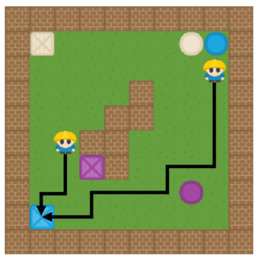

(c) Trajectories for disjunctive composition (collect blue or purple objects).

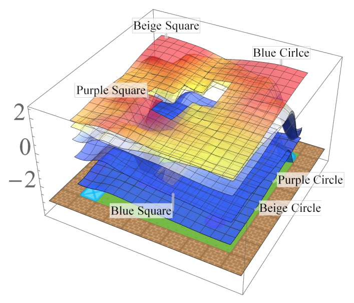


(d) Extended value function for conjunctive composition.

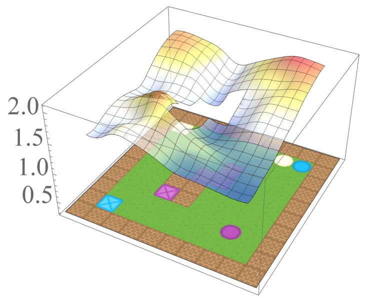


(e) Value function for conjunctive composition.


(f) Trajectories for conjunctive composition (collect blue squares).



(g) Extended value function for exclusive-or composition.



(h) Value function for exclusive-or composition.

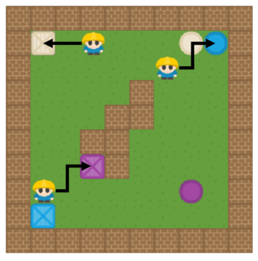

(i) Trajectories for exclusive-or composition (collect blue or square objects, but not blue squares).

Figure 5.20: Results for the video game environment with relaxed assumptions. We generate extended value functions to solve the disjunction of blue and purple tasks, and the conjunction and exclusive-or of blue and square tasks. The extended value functions shows the value function with respect to each goal plotted on the same axis. To generate this, we place the agent at every location and apply the element wise composition operators on the outputs of each network. We then compute the maximum over all actions for each goal. The points are interpolate between each other to smooth the graph. Any error in the visualisation is due to the use of non-linear function approximation with the relaxed assumptions.

Chapter 6

Future Work and Conclusion

6.1 Future Work

There is much room for improvement in learning the extended value functions. In our experiments, we learned each extended value function from scratch, but it is likely that having learned one for the first task, we could use it to initialise the extended value function for the second task to improve convergence times. One area for improvement lies in efficiently learning the extended value functions, as well as developing better algorithms for solving tasks with sparse rewards. For example, it is likely that approaches such as hindsight experience replay (Andrychowicz *et al.*, 2017) could reduce the number of samples required to learn extended value functions, while Mirowski *et al.* (2017) provides a method for learning complex tasks with sparse rewards using auxiliary tasks. We leave incorporating these approaches to future work.

Additionally, our experiments only considered simple domains as we placed more focus on the theoretical foundations. Future work could explore more complex domains involving difficult problems like robotic control, where learning is difficult and so minimal amount of learning is desired. While the theoretical results presented here easily extends to infinite MDPs with continuous state and action spaces, it is necessary to see if that holds in practice. This would potentially require more capable learning methods and function approximators for EVFs. They could perhaps take advantage of the rich knowledge (goal oriented values) that are encoded by EVFs. This is another exciting avenue for future work, especially given how EVFs are similar to population based methods that learn a diversity solutions for each task.

Regarding compositionality, we only considered concurrent composition. One exciting avenue for future work could be to extend the logical compositions presented here to temporal compositions. This could potentially be done by also following an algebraic approach, formalising temporal logics by using linear temporal logic operators on the established task space. Another method could simply be to combine the composition shown here with existing temporal composition methods like hierarchical reinforcement learning or options. It would give us the ability to solve zero-shot tasks like *collect blue squares then circles but never beige ones.* This is a much desired property for lifelong learning agents.

Another natural avenue for future work is to extend the composition framework beyond the classes of MDPs considered. These could be MDPs with different internal rewards, different terminal states, different environments (different state space, action space, transition dynamics), non-deterministic transition dynamics, partially observable

MDPs (POMDPs), and much more. POMDPs are a particularly interesting one since real world agents do not get to observe the entire state space. Extending zero-shot composition to them would hence make the compositions presented in this work much more applicable practically. Theoretical guarantees on such extensions would be ideal, but future work could also investigate practical methods of attaining the desired zero-shot composition.

Finally, beyond zero-shot learning and base tasks, we briefly discussed how regardless of the size of a task space the isomorphism results in Section 5.3 gives us theoretically the solution to any task by just learning the bounds of the value function space. Future work could look at efficient practical methods for computing that isomorphism given the set of desirable goals. Such methods are necessary in the case of infinite or even uncountable goal spaces, where generating EVFs using Boolean tables becomes impossible. One possible line of work could be to investigate using the isomorphism as a supervisor to train an agent using supervised learning on the true EVF rather than reinforcement learning on the MDP of the task. Any one of these extensions would help get us closer to the ultimate goal of lifelong learning agents in practice.

6.2 Conclusion

We have formalised the logical composition of tasks and value functions using lattice algebras, which are the algebraic structures of logics. This enables tasks to be treated algebraically in a similar way to sets in set theory, propositions in propositional logics, and digital signals in digital electronics. We then introduced extended value functions, which is a new type of value function for goal oriented tasks that encodes more information about solved tasks than standard value functions. We showed how an agent can learn a diversity of solutions for each task using it. This rich knowledge was shown to be sufficient to immediately solve composite tasks. This was done by first learning the extended value functions, and then composing them similarly to the composed task. This zero-shot ability combined with the notion of base tasks obtained from the task algebra guaranteed a super-exponential explosion of skills.

Our proposed approach is hence a step towards both interpretable RL—since both the tasks and optimal value functions can be specified using logic operators—and the ultimate goal of lifelong learning agents, which are able to solve combinatorially many tasks in a sample-efficient manner.

References

Abel, D., Jinnai, Y., Guo, S.Y., Konidaris, G., & Littman, M. 2018. Policy and Value Transfer in Lifelong Reinforcement Learning. *Pages 20–29 of: Proceedings of the 35th International Conference on Machine Learning.* Proceedings of Machine Learning Research, vol. 80. PMLR.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., & Zaremba, W. 2017. Hindsight experience replay. *Pages 5048–5058 of: Advances in Neural Information Processing Systems.*

Barreto, A., Dabney, W., Munos, R., Hunt, J., Schaul, T., van Hasselt, H., & Silver, D. 2017. Successor features for transfer in reinforcement learning. *Pages 4055–4065 of: Advances in neural information processing systems.*

Bellman, R. 1954. *The theory of dynamic programming.* Technical report. Rand corp santa monica ca.

Bertsekas, D.P., & Tsitsiklis, J.N. 1991. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, **16**(3), 580–595.

Birkhoff, G. 1940. *Lattice theory.* Vol. 25. American Mathematical Soc.

Boole, G. 1854. *An investigation of the laws of thought: on which are founded the mathematical theories of logic and probabilities.* Dover Publications.

Comtet, L. 2012. *Advanced Combinatorics: The art of finite and infinite expansions.* Springer Science & Business Media.

Foster, D., & Dayan, P. 2002. Structure in the space of value functions. *Machine Learning*, **49**(2-3), 325–346.

Fox, R., Pakman, A., & Tishby, N. 2016. Taming the noise in reinforcement learning via soft updates. *In: 32nd Conference on Uncertainty in Artificial Intelligence.*

Fu, M. C. 2016. AlphaGo and Monte Carlo tree search: the simulation optimization perspective. *Pages 659–670 of: Proceedings of the 2016 Winter Simulation Conference.* IEEE Press.

Grätzer, G. 2002. *General lattice theory.* Springer Science & Business Media.

Grätzer, G. 2011. *Lattice theory: foundation.* Springer Science & Business Media.

Haarnoja, T., Pong, V., Zhou, A., Dalal, M., Abbeel, P., & Levine, S. 2018. Composable Deep Reinforcement Learning for Robotic Manipulation. *arXiv preprint arXiv:1803.06773.*

Hunt, J., Barreto, A., Lillicrap, T., & Heess, N. 2019. Composing Entropic Policies using Divergence Correction. *Pages 2911–2920 of: Proceedings of the 36th International Conference on Machine Learning.* Proceedings of Machine Learning Research, vol. 97. PMLR.

Jaksch, T., Ortner, R., & Auer, P. 2010. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, **11**(Apr), 1563–1600.

James, H.W., & Collins, E.J. 2006. An analysis of transient Markov decision processes. *Journal of applied probability*, **43**(3), 603–621.

Kaelbling, Leslie Pack. 1993. Learning to achieve goals. *Pages 1094–1099 of: International Joint Conferences on Artificial Intelligence.*

Klir, G., & Yuan, B. 1995. *Fuzzy sets and fuzzy logic.* Vol. 4. Prentice hall New Jersey.

Levine, S., Finn, C., Darrell, T., & Abbeel, P. 2016. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, **17**(1), 1334–1373.

Lillicrap, T.., Hunt, J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. 2016. Continuous control with deep reinforcement learning. *In: International Conference on Learning Representations.*

Mirowski, P., Pascanu, R., Viola, F., Soyer, H., Ballard, A., Banino, A., Denil, M., Goroshin, R., Sifre, L., Kavukcuoglu, K., *et al.* 2017. Learning to navigate in complex environments. *In: International Conference on Learning Representations.*

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602.*

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A., Veness, J., Bellemare, M., Graves, A., Riedmiller, M., Fidjeland, A., Ostrovski, G., *et al.* 2015. Human-level control through deep reinforcement learning. *Nature*, **518**(7540), 529.

Monteiro, A. 1974. Matrices de Morgan Caracteristiques Pour le Calcul Propositionnel Classique.

Moore, A., Baird, L., & Kaelbling, L. 1999. Multi-value-functions: E cient automatic action hierarchies for multiple goal MDPs. *Pages 1316–1323 of: Proceedings of the international joint conference on artificial intelligence.*

Peng, X., Chang, M., Zhang, G., Abbeel, P., & Levine, S. 2019. MCP: Learning Composable Hierarchical Control with Multiplicative Compositional Policies. *arXiv preprint arXiv:1905.09808.*

Peng, X. B., Andrychowicz, M., Zaremba, W., & Abbeel, P. 2018. Sim-to-real transfer of robotic control with dynamics randomization. *Pages 1–8 of: 2018 IEEE International Conference on Robotics and Automation (ICRA).* IEEE.

Puterman, M.L. 2014. *Markov decision processes: discrete stochastic dynamic programming.* John Wiley & Sons.

Saxe, A.M., Earle, A.C., & Rosman, B.S. 2017. Hierarchy Through Composition with Multitask LMDPs. *Proceedings of the 34th International Conference on Machine Learning*, **70**, 3017–3026.

Schaul, T., Horgan, D., Gregor, K., & Silver, D. 2015. Universal Value Function Approximators. *Pages 1312–1320 of: Proceedings of the 32nd International Conference on Machine Learning*. Proceedings of Machine Learning Research, vol. 37. Lille, France: PMLR.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., *et al.* 2017. Mastering the game of go without human knowledge. *Nature*, **550**(7676), 354.

Sutton, R., Precup, D., & Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, **112**(1-2), 181–211.

Sutton, R., Modayil, J., Delp, M., Degris, T., Pilarski, P., White, A., & Precup, D. 2011. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. *Pages 761–768 of: The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*. International Foundation for Autonomous Agents and Multiagent Systems.

Sutton, Richard S, Barto, Andrew G, *et al.* 1998. *Introduction to reinforcement learning*. Vol. 135. MIT press Cambridge.

Thrun, S. 1996. Is learning the n-th thing any easier than learning the first? *Pages 640–646 of: Advances in neural information processing systems*.

Todorov, E. 2007. Linearly-solvable Markov decision problems. *Pages 1369–1376 of: Advances in Neural Information Processing Systems*.

Todorov, E. 2009. Compositionality of optimal control laws. *Pages 1856–1864 of: Advances in Neural Information Processing Systems*.

Van Niekerk, B., James, S., Earle, A., & Rosman, B. 2019. Composing Value Functions in Reinforcement Learning. *Pages 6401–6409 of: Proceedings of the 36th International Conference on Machine Learning*. Proceedings of Machine Learning Research, vol. 97. PMLR.

Veeriah, V., Oh, J., & Singh, S. 2018. Many-goals reinforcement learning. *arXiv preprint arXiv:1806.09605*.

Watkins, C. 1989. *Learning from delayed rewards*. Ph.D. thesis, King's College, Cambridge.

White, R. 1959. Motivation reconsidered: The concept of competence. *Psychological review*, **66**(5), 297.

Yen, J., & Langari, R. 1999. *Fuzzy logic: intelligence, control, and information*. Vol. 1. Prentice Hall Upper Saddle River, NJ.