# Chapter 3: Stochastic Processes + Bellman Foundations

## Contents

# 1 Chapter 3 — Stochastic Processes and Bellman Foundations

*Vlad Prytula*

## 1.1 3.1 Motivation: From Single Queries to Sequential Sessions

Chapter 1 formalized search ranking as a contextual bandit: observe context $x$ (user segment, query type), select action $a$ (boost weights), and observe an immediate reward $R(x, a, \omega)$. This abstraction is appropriate when each query can be treated as an independent decision, and when myopic objectives (GMV, CM2, clicks) are sufficient proxies for long-run value.

In deployed systems, user behavior is sequential. Actions taken on one query influence the distribution of future queries, clicks, and purchases within a session, and they can shape return probability across sessions. A user may refine a query after inspecting a ranking; a cart may accumulate over several steps; satisfaction may drift and eventually trigger abandonment. These are precisely the phenomena that a single-step model cannot represent.

Mathematically, the missing ingredient is **state**: a variable $S_t$ that summarizes the relevant history at time $t$ (cart, browsing context, latent satisfaction, recency). Once we represent the interaction as a controlled stochastic process $(S_0, A_0, R_0, S_1, A_1, R_1, ...)$, the central objects of reinforcement learning become well-defined:

- **Value functions** $V^\pi(s)$ and $Q^\pi(s, a)$, which measure expected cumulative reward under a policy $\pi$
- **Bellman operators** $\mathcal{T}^\pi$ and $\mathcal{T}$, which encode the dynamic programming principle as fixed-point equations

The guiding question of this chapter is structural: under what assumptions does repeated Bellman backup converge, and why does it converge to the optimal value function? The answer is an operator-theoretic one: the discounted Bellman operator is a contraction in the sup-norm, so it has a unique fixed point and value iteration converges to it.

We develop these foundations in the following order:

1. Section 3.2–3.3: Stochastic processes, filtrations, stopping times (measure-theoretic rigor for sequential randomness)
2. Section 3.4: Markov Decision Processes (formal definition, standard Borel assumptions)
3. Section 3.5: Bellman operators and value functions (from intuition to operators on function spaces)
4. Section 3.6: Contraction mappings and Banach fixed-point theorem (complete proof, step-by-step)
5. Section 3.7: Value iteration convergence (why dynamic programming works)
6. Section 3.8: Connection to bandits (the $\gamma = 0$ special case from Chapter 1)
7. Section 3.9: Computational verification (NumPy experiments)
8. Section 3.10: RL bridges (preview of Chapter 11's multi-episode formulation)

By the end of this chapter, we understand:

- Why value iteration converges exponentially fast (contraction mapping theorem)
- How to prove convergence of RL algorithms rigorously (fixed-point theory)
- When the bandit formulation is sufficient and when MDPs are necessary
- The mathematical foundations that justify TD-learning, Q-learning, and policy gradients

Prerequisites. This chapter assumes:

- Measure-theoretic probability from Chapter 2 (probability spaces, random variables, conditional expectation)
- Familiarity with supremum norm and function spaces (we will introduce contraction mappings from first principles)

---

## 1.2 3.2 Stochastic Processes: Modeling Sequential Randomness

**Definition 3.2.1** (Stochastic Process)

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, $T \subseteq \mathbb{R}_+$ an index set (often $T = \mathbb{N}$ or $T = [0, \infty)$), and $(E, \mathcal{E})$ a measurable space. A **stochastic process** is a collection of random variables $\{X_t : t \in T\}$ where each $X_t : \Omega \to E$ is $(\mathcal{F}, \mathcal{E})$-measurable.

**Notation**: We write $(X_t)_{t \in T}$ or simply $(X_t)$ when $T$ is clear from context.

**Intuition**: A stochastic process is a **time-indexed family of random variables**. Each $X_t$ represents the state of a system at time $t$. For a fixed $\omega \in \Omega$, the mapping $t \mapsto X_t(\omega)$ is a **sample path** or **trajectory**.

**Example 3.2.1** (User satisfaction process). Let $E = [0, 1]$ represent satisfaction levels. Define $S_t : \Omega \to [0, 1]$ as the user's satisfaction after the $t$-th query in a session. Then $(S_t)_{t=0}^{T}$ is a stochastic process modeling satisfaction evolution.

**Example 3.2.2** (RL trajectory). In a Markov Decision Process, the sequence $(S_0, A_0, R_0, S_1, A_1, R_1, \ldots)$ is a stochastic process where: - $S_t \in \mathcal{S}$ (state space) - $A_t \in \mathcal{A}$ (action space) - $R_t \in \mathbb{R}$ (reward)

Each component is a random variable, and their joint distribution is induced by the policy $\pi$ and environment dynamics $P$.

**Standing convention (Discrete time).** Throughout this chapter we work in discrete time with index set $T = \mathbb{N}$. Continuous-time analogues require additional measurability notions (e.g., predictable processes and optional $\sigma$-algebras), which we do not pursue here.

---

### 1.2.1 3.2.1 Filtrations and Adapted Processes

**Definition 3.2.2** (Filtration)

A **filtration** on $(\Omega, \mathcal{F}, \mathbb{P})$ is a collection $(\mathcal{F}_t)_{t\in T}$ of sub-$\sigma$-algebras of $\mathcal{F}$ satisfying:

$$\mathcal{F}_s \subseteq \mathcal{F}_t \subseteq \mathcal{F} \quad \text{for all } s \leq t.$$

**Intuition**: $\mathcal{F}_t$ represents the **information available at time** $t$. The inclusion $\mathcal{F}_s \subseteq \mathcal{F}_t$ captures the idea that information accumulates over time: we never "forget" past observations.

**Example 3.2.3** (Natural filtration). Given a stochastic process $(X_t)$, the **natural filtration** is:

$$\mathcal{F}_t := \sigma(X_s : s \leq t),$$

the smallest $\sigma$-algebra making all $X_s$ with $s \leq t$ measurable. This represents "all information revealed by observing $(X_0, X_1, \dots, X_t)$."

We write

$$\mathcal{F}_\infty := \sigma\left(\bigcup_{t\in T} \mathcal{F}_t\right)$$

for the terminal $\sigma$-algebra generated by the filtration.

**Definition 3.2.3** (Adapted Process)

A stochastic process $(X_t)$ is **adapted** to the filtration $(\mathcal{F}_t)$ if $X_t$ is $\mathcal{F}_t$-measurable for all $t \in T$.

**Intuition**: Adaptedness means "the value of $X_t$ is determined by information available at time $t$." This is the mathematical formalization of **causality**: $X_t$ cannot depend on future information $\mathcal{F}_s$ with $s > t$.

**Remark 3.2.1** (Adapted vs. predictable). In continuous-time stochastic calculus, there is a stronger notion called **predictable** (measurable with respect to $\mathcal{F}_{t-}$, the left limit). For discrete-time RL, adapted suffices.

**Remark 3.2.2** (RL policies must be adapted). In reinforcement learning, a policy $\pi(a|h_t)$ at time $t$ must depend only on the **history** $h_t = (s_0, a_0, r_0, \dots, s_t)$ available at $t$, not on future states $s_{t+1}, s_{t+2}, \dots$. This is precisely the adaptedness condition: $\pi_t$ is $\mathcal{F}_t$-measurable where $\mathcal{F}_t = \sigma(h_t)$.

---

### 1.2.2 3.2.2 Stopping Times

**Definition 3.2.4** (Stopping Time)

Let $(\mathcal{F}_t)$ be a filtration on $(\Omega, \mathcal{F}, \mathbb{P})$. A random variable $\tau : \Omega \to T \cup \{\infty\}$ is a **stopping time** if:

$$\{\tau \leq t\} \in \mathcal{F}_t \quad \text{for all } t \in T.$$

**Intuition**: A stopping time is a **random time** whose occurrence is determined by information available *up to that time*. The event "$\tau$ has occurred by time $t$" must be $\mathcal{F}_t$-measurable—we can decide whether to stop using only observations $(X_0, \dots, X_t)$, without peeking into the future.

**Example 3.2.4** (Session abandonment). Define:

$$\tau := \inf\{t \geq 0 : S_t < \theta\},$$

the first time user satisfaction $S_t$ drops below threshold $\theta$. This is a stopping time: to check "$\tau \leq t$" (user has abandoned by time $t$), we only need to observe $(S_0, \dots, S_t)$. We do not need to know future satisfaction $S_{t+1}, S_{t+2}, \dots$.

**Example 3.2.5** (Purchase event). Define $\tau$ as the first time the user makes a purchase. This is a stopping time: the event "$\tau = t$" means "user purchased at time $t$, having not purchased before"—determined by history up to $t$.

**Non-Example 3.2.6** (Last time satisfaction peaks). Define $\tau := \sup\{t : S_t = \max_{s \leq T} S_s\}$ (the last time satisfaction reaches its maximum over $[0, T]$). This is **NOT** a stopping time: to determine "$\tau = t$," we need to know future values $S_{t+1}, \dots, S_T$ to verify satisfaction never exceeds $S_t$ afterward.

**Proposition 3.2.1** (Measurability of stopped processes)

Assume $T = \mathbb{N}$. If $(X_t)$ is adapted to $(\mathcal{F}_t)$ and $\tau$ is a stopping time, then $X_\tau \mathbf{1}_{\{\tau < \infty\}}$ is $\mathcal{F}_\infty$-measurable.

*Proof.* **Step 1** (Indicator decomposition). Write:

$$X_\tau \mathbf{1}_{\{\tau<\infty\}} = \sum_{t=0}^{\infty} X_t \mathbf{1}_{\{\tau=t\}}.$$

**Step 2** (Measurability of indicators). For each $t \in \mathbb{N}$, the event $\{\tau = t\}$ belongs to $\mathcal{F}_t$. Indeed, $\{\tau = 0\} = \{\tau \leq 0\} \in \mathcal{F}_0$, and for $t \geq 1$,

$$\{\tau = t\} = \{\tau \leq t\} \cap \{\tau \leq t-1\}^c \in \mathcal{F}_t,$$

since $\{\tau \leq t\} \in \mathcal{F}_t$ and $\{\tau \leq t-1\} \in \mathcal{F}_{t-1} \subseteq \mathcal{F}_t$.

**Step 3** (Measurability of $X_t \mathbf{1}_{\{\tau=t\}}$). Since $X_t$ is $\mathcal{F}_t$-measurable and $\{\tau = t\} \in \mathcal{F}_t$, the product $X_t \mathbf{1}_{\{\tau=t\}}$ is $\mathcal{F}_t$-measurable, hence $\mathcal{F}_\infty$-measurable.

**Step 4** (Countable sum). A countable sum of measurable functions is measurable, so the right-hand side of Step 1 is $\mathcal{F}_\infty$-measurable; hence $X_\tau \mathbf{1}_{\{\tau<\infty\}}$ is $\mathcal{F}_\infty$-measurable. □

**Remark 3.2.3** (The indicator technique). The proof uses the **indicator decomposition**: write a stopped process as a sum over stopping events. This technique will reappear when proving optional stopping theorems for martingales (used in stochastic approximation convergence proofs, deferred to later chapters).

**Remark 3.2.4** (RL preview: Episode termination). In episodic RL, the terminal time $T$ is often a stopping time: the episode ends when the agent reaches a terminal state (e.g., user completes purchase or abandons session). The return $G_0 = \sum_{t=0}^{T-1} \gamma^t R_t$ depends on $T$, which is random. Proposition 3.2.1 ensures $G_0$ is well-defined as a random variable.

---

## 1.3 3.3 Markov Chains and the Markov Property

Before defining MDPs, we introduce **Markov chains**—stochastic processes with memoryless transitions.

**Definition 3.3.1** (Markov Chain)

A discrete-time stochastic process $(X_t)_{t \in \mathbb{N}}$ taking values in a countable or general measurable space $(E, \mathcal{E})$ is a **Markov chain** (with respect to its natural filtration $\mathcal{F}_t := \sigma(X_0, \dots, X_t)$) if:

$$\mathbb{P}(X_{t+1} \in A \mid \mathcal{F}_t) = \mathbb{P}(X_{t+1} \in A \mid X_t) \quad \text{for all } A \in \mathcal{E},\ t \geq 0. \tag{3.1}$$

This is the **Markov property**: the future $X_{t+1}$ is conditionally independent of the past $(X_0, \dots, X_{t-1})$ given the present $X_t$.

**Intuition**: "The future depends on the present, not on how we arrived at the present."

**Example 3.3.1** (Random walk). Let $(\xi_t)$ be i.i.d. random variables with $\mathbb{P}(\xi_t = +1) = \mathbb{P}(\xi_t = -1) = 1/2$. Define $X_t = \sum_{s=0}^{t-1} \xi_s$ (cumulative sum). Then:

$$X_{t+1} = X_t + \xi_t,$$

so $X_{t+1}$ depends only on $X_t$ and the new increment $\xi_t$ (independent of history). This is a Markov chain.

**Example 3.3.2** (User state transitions). In an e-commerce session, let $X_t \in \{\text{browsing}, \text{engaged}, \text{ready\_to\_buy}, \text{abandoned}\}$ be the user's state after $t$ queries. If transitions depend only on current state (not on the path taken to reach it), then $(X_t)$ is a Markov chain.

**Non-Example 3.3.3** (ARMA processes violate the Markov property). Consider $X_t = 0.5X_{t-1} + 0.3X_{t-2} + \varepsilon_t$ where $\varepsilon_t$ is white noise. Given only $X_t$, the distribution of $X_{t+1}$ depends on $X_{t-1}$ (through the 0.3 term), violating the Markov property (3.1). To restore Markovianity, **augment the state**: define $\tilde{X}_t = (X_t, X_{t-1})$. Then $\tilde{X}_{t+1}$ depends only on $\tilde{X}_t$, so $(\tilde{X}_t)$ is Markov. This **state augmentation** technique is fundamental in RL—frame stacking in video games (Remark 3.4.1), LSTM hidden states, and user history embeddings all restore the Markov property by expanding what we call "state."

**Definition 3.3.2** (Transition Kernel)

The **transition kernel** (or **transition probability**) of a Markov chain is:

$$P(x, A) := \mathbb{P}(X_{t+1} \in A \mid X_t = x), \quad x \in E,\, A \in \mathcal{E}.$$

For time-homogeneous chains, $P$ is independent of $t$.

**Properties**: 1. For each $x \in E$, $A \mapsto P(x, A)$ is a probability measure on $(E, \mathcal{E})$ 2. For each $A \in \mathcal{E}$, $x \mapsto P(x, A)$ is measurable

**Remark 3.3.1** (Standard Borel assumption). For general state spaces, we require $E$ to be a **standard Borel space** (a measurable subset of a Polish space, i.e., separable complete metric space). This ensures:

- Transition kernels $P(x, A)$ are well-defined and measurable
- Regular conditional probabilities exist
- Optimal policies can be chosen measurably

All finite and countable spaces are standard Borel. $\mathbb{R}^n$ with Borel $\sigma$-algebra is standard Borel. This covers essentially all RL applications.

---

## 1.4 3.4 Markov Decision Processes: The RL Framework

**Definition 3.4.1** (Markov Decision Process)

A **Markov Decision Process (MDP)** is a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ where:

1. $\mathcal{S}$ is the **state space** (a standard Borel space)
2. $\mathcal{A}$ is the **action space** (a standard Borel space)
3. $P(\cdot \mid s, a)$ is a **Markov kernel** from $(\mathcal{S} \times \mathcal{A}, \mathcal{B}(\mathcal{S}) \otimes \mathcal{B}(\mathcal{A}))$ to $(\mathcal{S}, \mathcal{B}(\mathcal{S}))$: $P(B \mid s, a)$ is the probability of transitioning to a Borel set $B \subseteq \mathcal{S}$ when taking action $a$ in state $s$
4. $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the **reward function**: $R(s, a, s')$ is the reward obtained from transition $(s, a, s')$
5. $\gamma \in [0, 1)$ is the **discount factor**

**Structural assumptions**: - For each $(s, a)$, $B \mapsto P(B \mid s, a)$ is a probability measure on $(\mathcal{S}, \mathcal{B}(\mathcal{S}))$ - For each $B \in \mathcal{B}(\mathcal{S})$, $(s, a) \mapsto P(B \mid s, a)$ is measurable - $R$ is bounded and measurable: $|R(s, a, s')| \leq R_{\max} < \infty$ for all $(s, a, s')$

**Notation**: - We write the bounded measurable one-step expected reward as

$$r(s, a) := \int_{\mathcal{S}} R(s, a, s')\, P(ds' \mid s, a).$$

When $\mathcal{S}$ is finite, integrals against $P(\cdot \mid s, a)$ reduce to sums: $\int_{\mathcal{S}} f(s')\, P(ds' \mid s, a) = \sum_{s' \in \mathcal{S}} P(s' \mid s, a) f(s')$.
- When $\mathcal{S}$ and $\mathcal{A}$ are finite, we represent $P$ as a tensor $P \in [0, 1]^{|\mathcal{S}| \times |\mathcal{A}| \times |\mathcal{S}|}$ with $P_{s,a,s'} = P(s' \mid s, a)$

**Definition 3.4.2** (Policy)

A **(stationary Markov) policy** is a **stochastic kernel** $\pi(\cdot \mid s)$ from $(\mathcal{S}, \mathcal{B}(\mathcal{S}))$ to $(\mathcal{A}, \mathcal{B}(\mathcal{A}))$ such that: - For each $s \in \mathcal{S}$, $B \mapsto \pi(B \mid s)$ is a probability measure on $(\mathcal{A}, \mathcal{B}(\mathcal{A}))$ - For each $B \in \mathcal{B}(\mathcal{A})$, $s \mapsto \pi(B \mid s)$ is $\mathcal{B}(\mathcal{S})$-measurable

We write integrals against the policy as $\pi(da \mid s)$. When $\mathcal{A}$ is finite, $\pi(a \mid s)$ is a mass function and $\int_{\mathcal{A}} f(a)\,\pi(da \mid s) = \sum_{a\in\mathcal{A}} f(a)\,\pi(a \mid s)$.

**Deterministic policies**: A policy is deterministic if $\pi(\cdot \mid s)$ is a point mass for all $s$. We identify deterministic policies with measurable functions $\pi : \mathcal{S} \to \mathcal{A}$, with the induced kernel $\pi(da \mid s) = \delta_{\pi(s)}(da)$.

**Assumption 3.4.1** (Markov assumption).

The MDP satisfies: 1. **Transition Markov property**: $\mathbb{P}(S_{t+1} \in B \mid s_0, a_0, \dots, s_t, a_t) = P(B \mid s_t, a_t)$ 2. **Reward Markov property**: $\mathbb{E}[R_t \mid s_0, a_0, \dots, s_t, a_t, s_{t+1}] = R(s_t, a_t, s_{t+1})$

These properties ensure that **state $s_t$ summarizes all past information relevant for predicting the future**. This is crucial: if the state does not satisfy the Markov property, the MDP framework breaks down.

**Remark 3.4.1** (State design in practice). Real systems rarely have perfectly Markovian observations. Practitioners construct **augmented states** to restore the Markov property:

- **Frame stacking** in video games: stack last 4 frames to capture velocity
- **LSTM hidden states**: recurrent network state becomes part of MDP state
- **User history embeddings**: include session features (past clicks, queries) in context vector

This is a modeling choice rather than a theorem, but it is essential for applying MDP theory in practice.

---

### 1.4.1 3.4.1 Value Functions

**Definition 3.4.3** (State-Value Function)

Given a policy $\pi$ and initial state $s \in \mathcal{S}$, the **state-value function** is:

$$V^\pi(s) := \mathbb{E}^\pi\left[\sum_{t=0}^{\infty} \gamma^t R_t \,\middle|\, S_0 = s\right], \tag{3.2}$$

where the expectation is over trajectories $(S_0, A_0, R_0, S_1, A_1, R_1, \dots)$ generated by policy $\pi$ and transition kernel $P$: - $S_0 = s$ (initial state) - $A_t \sim \pi(\cdot|S_t)$ (actions sampled from policy) - $S_{t+1} \sim P(\cdot|S_t, A_t)$ (states transition according to dynamics) - $R_t = R(S_t, A_t, S_{t+1})$ (rewards realized from transitions)

**Notation**: The superscript $\mathbb{E}^\pi$ emphasizes that the expectation is under the probability measure $\mathbb{P}^\pi$ induced by policy $\pi$ and dynamics $P$. Existence and uniqueness of this trajectory measure (built from the initial state, the policy kernel, and the transition kernel) can be formalized via the Ionescu–Tulcea extension theorem; Chapter 2 gives the measurable construction of MDP trajectories.

**Well-definedness**: Since $\gamma < 1$ and $|R_t| \le R_{\max}$, the series converges absolutely:

$$\left|\sum_{t=0}^{\infty} \gamma^t R_t\right| \le \sum_{t=0}^{\infty} \gamma^t R_{\max} = \frac{R_{\max}}{1-\gamma} < \infty.$$

Thus $V^\pi(s)$ is well-defined and $|V^\pi(s)| \le R_{\max}/(1-\gamma)$ for all $s, \pi$.

**Definition 3.4.4** (Action-Value Function)

Given a policy $\pi$, state $s$, and action $a$, the **action-value function** (or **Q-function**) is:

$$Q^\pi(s,a) := \mathbb{E}^\pi\left[\sum_{t=0}^{\infty} \gamma^t R_t \,\middle|\, S_0 = s, A_0 = a\right]. \tag{3.3}$$

This is the expected return starting from state $s$, taking action $a$, then following policy $\pi$.

**Relationship**:

$$V^\pi(s) = \int_{\mathcal{A}} Q^\pi(s,a)\,\pi(da \mid s), \tag{3.4}$$

When $\mathcal{A}$ is finite, the integral reduces to the familiar sum $\sum_{a\in\mathcal{A}} \pi(a \mid s) Q^\pi(s,a)$.

**Definition 3.4.5** (Optimal Value Functions)

The **optimal state-value function** is:

$$V^*(s) := \sup_\pi V^\pi(s), \tag{3.5}$$

and the **optimal action-value function** is:

$$Q^*(s,a) := \sup_\pi Q^\pi(s,a). \tag{3.6}$$

**Remark 3.4.2** (Existence of optimal policies and measurable selection). In finite action spaces, optimal actions exist statewise and the Bellman optimality operator can be written with max. In general Borel state-action spaces, the optimality operator is stated with sup, and existence of a **deterministic stationary** optimal policy can require additional topological conditions (e.g., compact $\mathcal{A}$ and upper semicontinuity) or a measurable selection theorem.

Chapter 2 discusses this fine print and gives a measurable formulation of the Bellman operators; see also (Puterman 2014, Theorem 6.2.10) for a comprehensive treatment. In this chapter, we focus on statements and proofs that do not require selecting maximizers: operator well-definedness on bounded measurable functions and contraction in $\|\cdot\|_\infty$.

---

## 1.5 3.5 Bellman Equations

The Bellman equations provide **recursive characterizations** of value functions. These are the cornerstone of RL theory.

**Theorem 3.5.1** (Bellman Expectation Equation)

For any policy $\pi$, the value function $V^\pi$ satisfies:

$$V^\pi(s) = \int_{\mathcal{A}} \left[ r(s,a) + \gamma \int_{\mathcal{S}} V^\pi(s')\,P(ds' \mid s,a) \right] \pi(da \mid s), \tag{3.7}$$

where $r(s,a) = \int_{\mathcal{S}} R(s,a,s')\,P(ds' \mid s,a)$ as in 3.4.1.

Equivalently, in operator notation:

$$V^\pi = \mathcal{T}^\pi V^\pi, \tag{3.8}$$

where $\mathcal{T}^\pi$ is the **Bellman expectation operator** for policy $\pi$:

$$(\mathcal{T}^\pi V)(s) := \int_{\mathcal{A}} \left[ r(s,a) + \gamma \int_{\mathcal{S}} V(s')\,P(ds' \mid s,a) \right] \pi(da \mid s). \tag{3.9}$$

*Proof.* **Step 1** (Decompose the return). By definition (3.2),

$$V^\pi(s) = \mathbb{E}^\pi \left[ \sum_{t=0}^\infty \gamma^t R_t \,\middle|\, S_0 = s \right].$$

Separate the first reward from the tail:

$$V^\pi(s) = \mathbb{E}^\pi \left[ R_0 + \gamma \sum_{t=1}^{\infty} \gamma^{t-1} R_t \,\middle|\, S_0 = s \right].$$

**Step 2** (Tower property). Apply the law of total expectation (tower property) conditioning on $(A_0, S_1)$:

$$V^\pi(s) = \mathbb{E}^\pi \left[ \mathbb{E}^\pi \left[ R_0 + \gamma \sum_{t=1}^{\infty} \gamma^{t-1} R_t \,\middle|\, S_0 = s, A_0, S_1 \right] \right].$$

**Step 3** (Markov property). Since $R_0 = R(S_0, A_0, S_1)$ is determined by $(S_0, A_0, S_1)$, and future rewards $(R_1, R_2, \ldots)$ depend only on $S_1$ onward (Markov property [ASM-3.4.1]), we have:

$$\mathbb{E}^\pi \left[ R_0 + \gamma \sum_{t=1}^{\infty} \gamma^{t-1} R_t \,\middle|\, S_0 = s, A_0 = a, S_1 = s' \right] = R(s, a, s') + \gamma V^\pi(s').$$

**Step 4** (Integrate over actions and next states). Taking expectations over $A_0 \sim \pi(\cdot \mid s)$ and $S_1 \sim P(\cdot \mid s, a)$:

$$V^\pi(s) = \int_{\mathcal{A}} \int_{\mathcal{S}} [R(s, a, s') + \gamma V^\pi(s')] \, P(ds' \mid s, a) \, \pi(da \mid s).$$

Rearranging:

$$V^\pi(s) = \int_{\mathcal{A}} \left[ \underbrace{\int_{\mathcal{S}} R(s, a, s') \, P(ds' \mid s, a)}_{=: r(s,a)} + \gamma \int_{\mathcal{S}} V^\pi(s') \, P(ds' \mid s, a) \right] \pi(da \mid s),$$

which is (3.7). □

**Remark 3.5.1** (The dynamic programming principle). The proof uses the **principle of optimality**: breaking the infinite-horizon return into immediate reward plus discounted future value. This is the essence of dynamic programming. The Markov property 3.4.1 is crucial—without it, $V^\pi(s')$ would depend on the history leading to $s'$, and the recursion would fail.

**Theorem 3.5.2** (Bellman Optimality Equation)

The optimal value function $V^*$ satisfies:

$$V^*(s) = \sup_{a \in \mathcal{A}} \left[ r(s, a) + \gamma \int_{\mathcal{S}} V^*(s') \, P(ds' \mid s, a) \right], \tag{3.10}$$

or in operator notation:

$$V^* = \mathcal{T} V^*, \tag{3.11}$$

where $\mathcal{T}$ is the **Bellman optimality operator**:

$$(\mathcal{T} V)(s) := \sup_{a \in \mathcal{A}} \left[ r(s, a) + \gamma \int_{\mathcal{S}} V(s') \, P(ds' \mid s, a) \right]. \tag{3.12}$$

*Proof.* **Step 1** (Control dominates evaluation). For any bounded measurable function $V$ and any policy $\pi$, we have, for each $s \in \mathcal{S}$,

$$(\mathcal{T}^\pi V)(s) = \int_{\mathcal{A}} \left[ r(s, a) + \gamma \int_{\mathcal{S}} V(s') \, P(ds' \mid s, a) \right] \pi(da \mid s) \le \sup_{a \in \mathcal{A}} \left[ r(s, a) + \gamma \int_{\mathcal{S}} V(s') \, P(ds' \mid s, a) \right] = (\mathcal{T} V)(s).$$

**Step 2** (Fixed point characterization of the optimal value). Section 3.7 shows that $\mathcal{T}$ is a $\gamma$-contraction on $(B_b(\mathcal{S}), \|\cdot\|_\infty)$, hence has a unique fixed point $\bar{V}$ by 3.6.2. Similarly, for each policy $\pi$, the evaluation operator $\mathcal{T}^\pi$ is a $\gamma$-contraction (the proof is the same as for 3.7.1, without the supremum), hence has a unique fixed point $V^\pi$.

By Step 1, $(\mathcal{T}^\pi)^k V \leq \mathcal{T}^k V$ for all $k$ and all $V \in B_b(\mathcal{S})$. Taking $k \to \infty$ yields $V^\pi \leq \bar{V}$, hence $\sup_\pi V^\pi \leq \bar{V}$ pointwise.

Discounted dynamic-programming theory shows that the fixed point $\bar{V}$ coincides with the optimal value function $V^*$ defined in (3.5), and therefore satisfies $V^* = \mathcal{T} V^*$; see (Puterman 2014, Theorem 6.2.10) for the measurable-selection details behind this identification. This gives (3.11) and the pointwise form (3.10). □

**Note on proof structure.** This proof invokes 3.7.1 (Bellman contraction) and 3.6.2 (Banach fixed-point), which we establish in Section 3.6–3.7. We state the optimality equation here because it is conceptually fundamental—*this is the equation RL algorithms solve.* The existence and uniqueness of $V^*$ follow once we prove $\mathcal{T}$ is a contraction in Section 3.7.

**Remark 3.5.2** (Suprema, maximizers, and greedy policies). Equation (3.10) is stated with sup because the supremum need not be attained without additional assumptions. In finite action spaces, or under compactness/upper-semicontinuity conditions, the supremum is attained and we may write max.

When a measurable maximizer exists, we can extract a deterministic greedy policy via:

$$\pi^*(s) \in \arg\max_{a \in \mathcal{A}} \left[ r(s,a) + \gamma \int_{\mathcal{S}} V^*(s')\, P(ds' \mid s,a) \right].$$

Without measurable maximizers, we work with $\varepsilon$-optimal selectors and interpret (3.10) as a value characterization; Chapter 2 discusses the measurable-selection fine print in more detail.

**Remark 3.5.3** (CMDPs and regret: where the details live). Many practical ranking problems impose constraints (e.g., CM2 floors, exposure parity). **Constrained MDPs** (CMDPs) handle these by introducing Lagrange multipliers that convert the constrained problem into an unconstrained MDP with modified rewards $r_\lambda = r - \lambda c$—the Bellman theory of this section then applies directly to the relaxed problem. Appendix C develops the full CMDP framework with rigorous duality and algorithms; see THM-C.2.1, COR-C.3.1, and [ALG-C.5.1]. Chapter 10 treats constraints operationally as production guardrails, while Chapter 14 implements soft constraint optimization via primal–dual methods.

Regret guarantees are developed in Chapter 6 (e.g., THM-6.1, [THM-6.2]) with information-theoretic lower bounds in Appendix D ([THM-D.3.1]).

Once we compute $V^*$ (via value iteration, which we will prove converges next), extracting the optimal policy is straightforward.

---

## 1.6 3.6 Contraction Mappings and the Banach Fixed-Point Theorem

The Bellman operator $\mathcal{T}$ is a **contraction mapping**. This fundamental property guarantees: 1. Existence and uniqueness of the fixed point $V^* = \mathcal{T} V^*$ 2. Convergence of value iteration: $V_{k+1} = \mathcal{T} V_k \to V^*$ exponentially fast

We now develop this theory rigorously.

### 1.6.1 3.6.1 Normed Spaces and Contractions

**Definition 3.6.1** (Normed Vector Space)

A **normed vector space** is a pair $(V, \|\cdot\|)$ where $V$ is a vector space (over $\mathbb{R}$ or $\mathbb{C}$) and $\|\cdot\| : V \to \mathbb{R}_+$ is a **norm** satisfying: 1. **Positive definiteness**: $\|v\| = 0 \iff v = 0$ 2. **Homogeneity**: $\|\alpha v\| = |\alpha| \|v\|$ for all scalars $\alpha$ 3. **Triangle inequality**: $\|u + v\| \leq \|u\| + \|v\|$ for all $u, v \in V$

**Definition 3.6.2** (Supremum Norm)

For bounded measurable functions $f : \mathcal{S} \to \mathbb{R}$, the **supremum norm** (or $\infty$**-norm**) is:

$$\|f\|_\infty := \sup_{s \in \mathcal{S}} |f(s)|. \tag{3.13}$$

We write $B_b(\mathcal{S})$ for the space of bounded measurable functions:

$$B_b(\mathcal{S}) := \{f : \mathcal{S} \to \mathbb{R} \text{ measurable} : \|f\|_\infty < \infty\}.$$

Then $(B_b(\mathcal{S}), \|\cdot\|_\infty)$ is a normed vector space.

**Proposition 3.6.1** (Completeness of $(B_b(\mathcal{S}), \|\cdot\|_\infty)$)

The space $(B_b(\mathcal{S}), \|\cdot\|_\infty)$ is **complete**: every Cauchy sequence converges.

*Proof.* **Step 1** (Cauchy implies pointwise Cauchy). Let $(f_n)$ be a Cauchy sequence in $B_b(\mathcal{S})$. For each $s \in \mathcal{S}$,

$$|f_n(s) - f_m(s)| \le \|f_n - f_m\|_\infty \to 0 \quad \text{as } n, m \to \infty.$$

Thus $(f_n(s))$ is a Cauchy sequence in $\mathbb{R}$. Since $\mathbb{R}$ is complete, $f_n(s) \to f(s)$ for some $f(s) \in \mathbb{R}$.

**Step 2** (Uniform boundedness and measurability). Since $(f_n)$ is Cauchy, it is bounded: $\sup_n \|f_n\|_\infty \le M < \infty$. Thus $|f(s)| = \lim_n |f_n(s)| \le M$ for all $s$, so $\|f\|_\infty \le M < \infty$. Since each $f_n$ is measurable and $f_n \to f$ pointwise, the limit $f$ is measurable.

**Step 3** (Uniform convergence). Given $\epsilon > 0$, choose $N$ such that $\|f_n - f_m\|_\infty < \epsilon$ for all $n, m \ge N$. Fixing $n \ge N$ and taking $m \to \infty$:

$$|f_n(s) - f(s)| = \lim_{m \to \infty} |f_n(s) - f_m(s)| \le \epsilon \quad \text{for all } s.$$

Thus $\|f_n - f\|_\infty \le \epsilon$ for all $n \ge N$, proving $f_n \to f$ in $\|\cdot\|_\infty$. □

**Remark 3.6.1** (Banach spaces and uniform convergence). A complete normed space is called a **Banach space**. Proposition 3.6.1 shows $B_b(\mathcal{S})$ is a Banach space—this is essential for applying the Banach fixed-point theorem.

A crucial subtlety: Step 3 establishes **uniform convergence**, where $\sup_s |f_n(s) - f(s)| \to 0$. This is strictly stronger than **pointwise convergence** (where each $f_n(s) \to f(s)$ individually, which Step 1 provides). The space of bounded functions is complete under uniform convergence but *not* under pointwise convergence—a sequence of bounded continuous functions can converge pointwise to an unbounded or discontinuous function. This distinction matters: the Banach fixed-point theorem requires completeness in the norm topology, and value iteration convergence guarantees 3.7.3 are statements about uniform convergence over all states.

**Definition 3.6.3** (Contraction Mapping)

Let $(V, \|\cdot\|)$ be a normed space. A mapping $T : V \to V$ is a $\gamma$**-contraction** if there exists $\gamma \in [0, 1)$ such that:

$$\|T(f) - T(g)\| \le \gamma \|f - g\| \quad \text{for all } f, g \in V. \tag{3.14}$$

**Intuition**: $T$ brings points closer together by a factor $\gamma < 1$. The distance between $T(f)$ and $T(g)$ is strictly smaller than the distance between $f$ and $g$ (unless $f = g$).

**Example 3.6.1** (Scalar contraction). Let $V = \mathbb{R}$ with norm $|x|$. Define $T(x) = \frac{1}{2}x + 1$. Then:

$$|T(x) - T(y)| = \left|\frac{1}{2}(x - y)\right| = \frac{1}{2}|x - y|,$$

so $T$ is a 1/2-contraction.

**Non-Example 3.6.2** (Expansion). Define $T(x) = 2x$. Then $|T(x) - T(y)| = 2|x - y|$, so $T$ is an **expansion**, not a contraction.

---

### 1.6.2 3.6.2 Banach Fixed-Point Theorem

**Theorem 3.6.2** (Banach Fixed-Point Theorem)

Let $(V, \|\cdot\|)$ be a complete normed space (Banach space) and $T : V \to V$ a $\gamma$-contraction with $\gamma \in [0, 1)$. Then:

1. **Existence**: $T$ has a **unique fixed point** $v^* \in V$ satisfying $T(v^*) = v^*$
2. **Convergence**: For any initial point $v_0 \in V$, the sequence $v_{k+1} = T(v_k)$ converges to $v^*$
3. **Rate**: The convergence is **exponential**:

$$\|v_k - v^*\| \leq \frac{\gamma^k}{1-\gamma}\|T(v_0) - v_0\| \tag{3.15}$$

*Proof.* We prove each claim step-by-step.

**Proof of (1): Uniqueness** Suppose $T(v^*) = v^*$ and $T(w^*) = w^*$ are two fixed points. Then:

$$\|v^* - w^*\| = \|T(v^*) - T(w^*)\| \leq \gamma\|v^* - w^*\|.$$

Since $\gamma < 1$, this implies $\|v^* - w^*\| = 0$, hence $v^* = w^*$. $\square$ (Uniqueness)

**Proof of (2): Convergence to a fixed point Step 1** (Sequence is Cauchy). Define $v_k := T^k(v_0)$ (applying $T$ iteratively). For $k \geq 1$:

$$\|v_{k+1} - v_k\| = \|T(v_k) - T(v_{k-1})\| \leq \gamma\|v_k - v_{k-1}\|.$$

Iterating this inequality:

$$\|v_{k+1} - v_k\| \leq \gamma^k\|v_1 - v_0\|.$$

For $n > m$, by the triangle inequality:

$$\begin{aligned}
\|v_n - v_m\| &\leq \sum_{k=m}^{n-1}\|v_{k+1} - v_k\| && (1)\\
&\leq \sum_{k=m}^{n-1}\gamma^k\|v_1 - v_0\| && (2)\\
&= \gamma^m\frac{1-\gamma^{n-m}}{1-\gamma}\|v_1 - v_0\| && (3)\\
&\leq \frac{\gamma^m}{1-\gamma}\|v_1 - v_0\|. && (4)
\end{aligned}$$

Since $\gamma < 1$, $\gamma^m \to 0$ as $m \to \infty$, so $(v_k)$ is a Cauchy sequence.

**Step 2** (Completeness implies convergence). Since $V$ is complete, there exists $v^* \in V$ such that $v_k \to v^*$.

**Step 3** (Limit is a fixed point). Since $T$ is a contraction, it is continuous. Thus:

$$T(v^*) = T\left(\lim_{k\to\infty} v_k\right) = \lim_{k\to\infty} T(v_k) = \lim_{k\to\infty} v_{k+1} = v^*.$$

So $v^*$ is a fixed point. By uniqueness (proved above), it is the **unique** fixed point. $\square$ (Existence and Convergence)

**Proof of (3): Rate** From Step 1 above, taking $m = 0$ and letting $n \to \infty$:

$$\|v^* - v_0\| \leq \sum_{k=0}^{\infty}\|v_{k+1} - v_k\| \leq \|v_1 - v_0\|\sum_{k=0}^{\infty}\gamma^k = \frac{\|v_1 - v_0\|}{1-\gamma}.$$

For $k \geq 1$, applying the contraction property:

$$\|v_k - v^*\| = \|T(v_{k-1}) - T(v^*)\| \leq \gamma\|v_{k-1} - v^*\|.$$

Iterating:

$$\|v_k - v^*\| \leq \gamma^k\|v_0 - v^*\| \leq \frac{\gamma^k}{1-\gamma}\|v_1 - v_0\|,$$

which is EQ-3.15. □ (Rate)

**Remark 3.6.2** (The key mechanisms). This proof deploys several fundamental techniques:

1. **Telescoping series**: Write $\|v_n - v_m\| \leq \sum_{k=m}^{n-1} \|v_{k+1} - v_k\|$ to control differences
2. **Geometric series**: Bound $\sum_{k=m}^{\infty} \gamma^k = \gamma^m/(1-\gamma)$ using $\gamma < 1$
3. **Completeness**: Cauchy sequences converge—this is **essential** and fails in incomplete spaces (e.g., rationals $\mathbb{Q}$)
4. **Continuity from contraction**: Contractions are uniformly continuous, so limits pass through $T$

These techniques will reappear in convergence proofs for TD-learning (Chapters 8, 12) and stochastic approximation (later chapters).

**Example 3.6.3** (Failure without completeness). Define $T : \mathbb{Q} \to \mathbb{Q}$ by $T(x) = (x + 2/x)/2$—Newton-Raphson iteration for finding $\sqrt{2}$. Near $x = 1.5$, this map is a contraction: $|T(x) - T(y)| < 0.5|x - y|$ for $x, y \in [1, 2] \cap \mathbb{Q}$. Starting from $x_0 = 3/2 \in \mathbb{Q}$, the sequence $x_{k+1} = T(x_k)$ remains in $\mathbb{Q}$ and converges… but to $\sqrt{2} \notin \mathbb{Q}$. The fixed point exists in $\mathbb{R}$ but not in the incomplete space $\mathbb{Q}$. This is why completeness is essential for [THM-3.6.2-Banach]—and why we need $B_b(\mathcal{S})$ to be a Banach space for value iteration to converge to a *valid* value function.

**Remark 3.6.3** (The $1/(1-\gamma)$ factor). The bound EQ-3.15 shows the convergence rate depends on $1/(1-\gamma)$. When $\gamma \to 1$ (nearly undiscounted), convergence slows dramatically—this explains why high-$\gamma$ RL (e.g., $\gamma = 0.99$) requires many iterations. The factor $\gamma^k$ gives **exponential convergence**: doubling $k$ squares the error.

---

## 1.7 3.7 Bellman Operator is a Contraction

We now prove the central result: the Bellman optimality operator $\mathcal{T}$ is a $\gamma$-contraction on $(B_b(\mathcal{S}), \|\cdot\|_\infty)$.

**Remark 3.7.0** (Self-mapping property). Before proving contraction, we verify that $\mathcal{T}$ maps bounded measurable functions to bounded measurable functions. Under our standing assumptions—bounded rewards $|r(s,a)| \leq R_{\max}$ and discount $\gamma < 1$ ([DEF-3.4.1])—if $\|V\|_\infty < \infty$, then:

$$\|\mathcal{T}V\|_\infty = \sup_{s\in\mathcal{S}} \left|\sup_{a\in\mathcal{A}} \left[r(s,a) + \gamma \int_{\mathcal{S}} V(s')\, P(ds' \mid s,a)\right]\right| \leq R_{\max} + \gamma\|V\|_\infty < \infty.$$

This establishes boundedness. Measurability of $s \mapsto (\mathcal{T}V)(s)$ is immediate in the finite-action case (where the supremum is a maximum over finitely many measurable functions) and holds under standard topological hypotheses; see PROP-2.8.2 and Chapter 2, §2.8.2 (in particular [THM-2.8.3]).

**Theorem 3.7.1** (Bellman Operator Contraction)

The Bellman optimality operator $\mathcal{T} : B_b(\mathcal{S}) \to B_b(\mathcal{S})$ defined by:

$$(\mathcal{T}V)(s) = \sup_{a\in\mathcal{A}} \left[r(s,a) + \gamma \int_{\mathcal{S}} V(s')\, P(ds' \mid s,a)\right]$$

is a $\gamma$-contraction with respect to $\|\cdot\|_\infty$:

$$\|\mathcal{T}V - \mathcal{T}W\|_\infty \leq \gamma\|V - W\|_\infty \quad \text{for all } V, W \in B_b(\mathcal{S}). \tag{3.16}$$

*Proof.* **Step 1** (Non-expansiveness of sup). For any real-valued functions $f, g$ on $\mathcal{A}$,

$$\left|\sup_{a\in\mathcal{A}} f(a) - \sup_{a\in\mathcal{A}} g(a)\right| \leq \sup_{a\in\mathcal{A}} |f(a) - g(a)|.$$

Indeed, $f(a) \leq g(a) + |f(a) - g(a)| \leq \sup_{a'} g(a') + \sup_{a'} |f(a') - g(a')|$ for all $a$, so taking $\sup_a$ yields $\sup_a f(a) \leq \sup_a g(a) + \sup_a |f(a) - g(a)|$. Swapping $f, g$ gives the reverse inequality, and combining yields the claim.

**Step 2** (Pointwise contraction). Fix $s \in \mathcal{S}$ and define, for each $a \in \mathcal{A}$,

$$F_V(a) := r(s,a) + \gamma \int_{\mathcal{S}} V(s')\, P(ds' \mid s,a), \qquad F_W(a) := r(s,a) + \gamma \int_{\mathcal{S}} W(s')\, P(ds' \mid s,a).$$

Then $(\mathcal{T}V)(s) = \sup_a F_V(a)$ and $(\mathcal{T}W)(s) = \sup_a F_W(a)$. By Step 1,

$$|(\mathcal{T}V)(s) - (\mathcal{T}W)(s)| \leq \sup_{a\in\mathcal{A}} |F_V(a) - F_W(a)| = \gamma \sup_{a\in\mathcal{A}} \left|\int_{\mathcal{S}} (V-W)(s')\, P(ds' \mid s,a)\right|.$$

Since $P(\cdot \mid s,a)$ is a probability measure and $V - W$ is bounded,

$$\left|\int_{\mathcal{S}} (V-W)(s')\, P(ds' \mid s,a)\right| \leq \int_{\mathcal{S}} |V(s') - W(s')|\, P(ds' \mid s,a) \leq \|V - W\|_\infty.$$

Therefore $|(\mathcal{T}V)(s) - (\mathcal{T}W)(s)| \leq \gamma \|V - W\|_\infty$ for all $s$.

**Step 3** (Supremum over states). Taking $\sup_{s\in\mathcal{S}}$ yields $\|\mathcal{T}V - \mathcal{T}W\|_\infty \leq \gamma \|V - W\|_\infty$, which is (3.16). ☐

**Remark 3.7.1** (The sup-stability mechanism). The proof exploits the **non-expansiveness of** sup: taking a supremum is a 1-Lipschitz operation. Formally, for any functions $f, g$,

$$|\sup_a f(a) - \sup_a g(a)| \leq \sup_a |f(a) - g(a)|.$$

This is a fundamental technique in dynamic programming theory, appearing in proofs of policy improvement theorems and error propagation bounds.

**Remark 3.7.2** (Norm specificity). The contraction (3.16) holds specifically in the **sup-norm** $\|\cdot\|_\infty$. The Bellman operator is generally **not** a contraction in $L^1$ or $L^2$ norms—the proof crucially uses

$$\int_{\mathcal{S}} |V(s') - W(s')|\, P(ds' \mid s,a) \leq \|V - W\|_\infty,$$

which fails for other $L^p$ norms. This norm choice has practical implications: error bounds in RL propagate through the $\|\cdot\|_\infty$ norm, meaning worst-case state errors matter most.

**Corollary 3.7.2** (Existence and Uniqueness of $V^*$)

There exists a unique $V^* \in B_b(\mathcal{S})$ satisfying the Bellman optimality equation $V^* = \mathcal{T}V^*$.

*Proof.* Immediate from Theorems 3.6.2 and 3.7.1: $\mathcal{T}$ is a $\gamma$-contraction on the Banach space $(B_b(\mathcal{S}), \|\cdot\|_\infty)$, so it has a unique fixed point. ☐

**Corollary 3.7.3** (Value Iteration Convergence)

For any initial value function $V_0 \in B_b(\mathcal{S})$, the sequence:

$$V_{k+1} = \mathcal{T}V_k \tag{3.17}$$

converges to $V^*$ with exponential rate:

$$\|V_k - V^*\|_\infty \leq \frac{\gamma^k}{1-\gamma} \|\mathcal{T}V_0 - V_0\|_\infty. \tag{3.18}$$

*Proof.* Immediate from Theorems 3.6.2 and 3.7.1. □

**Proposition 3.7.4** (Reward perturbation sensitivity)

Fix $(\mathcal{S}, \mathcal{A}, P, \gamma)$ and let $r$ and $\tilde{r} = r + \Delta r$ be bounded measurable one-step reward functions. Let $V_r^*$ and $V_{\tilde{r}}^*$ denote the unique fixed points of the corresponding Bellman optimality operators on $B_b(\mathcal{S})$. Then:

$$\|V_{\tilde{r}}^* - V_r^*\|_\infty \le \frac{\|\Delta r\|_\infty}{1-\gamma}.$$

*Proof.* Let $\mathcal{T}_r$ and $\mathcal{T}_{\tilde{r}}$ denote the two Bellman optimality operators. For any $V \in B_b(\mathcal{S})$ and any $s \in \mathcal{S}$,

$$|(\mathcal{T}_{\tilde{r}}V)(s) - (\mathcal{T}_r V)(s)| = \left|\sup_{a\in\mathcal{A}} \left[\tilde{r}(s,a) + \gamma \int_{\mathcal{S}} V(s')\, P(ds' \mid s,a)\right] - \sup_{a\in\mathcal{A}} \left[r(s,a) + \gamma \int_{\mathcal{S}} V(s')\, P(ds' \mid s,a)\right]\right| \le \sup_{a\in\mathcal{A}} |\tilde{r}(s,a) - r(s,a$$

so $\|\mathcal{T}_{\tilde{r}}V - \mathcal{T}_r V\|_\infty \le \|\Delta r\|_\infty$.

Using the fixed point identities $V_r^* = \mathcal{T}_r V_r^*$ and $V_{\tilde{r}}^* = \mathcal{T}_{\tilde{r}} V_{\tilde{r}}^*$ and the contraction of $\mathcal{T}_{\tilde{r}}$,

$$\|V_{\tilde{r}}^* - V_r^*\|_\infty = \|\mathcal{T}_{\tilde{r}}V_{\tilde{r}}^* - \mathcal{T}_r V_r^*\|_\infty \le \|\mathcal{T}_{\tilde{r}}V_{\tilde{r}}^* - \mathcal{T}_{\tilde{r}}V_r^*\|_\infty + \|\mathcal{T}_{\tilde{r}}V_r^* - \mathcal{T}_r V_r^*\|_\infty \le \gamma\|V_{\tilde{r}}^* - V_r^*\|_\infty + \|\Delta r\|_\infty.$$

Rearranging yields $\|V_{\tilde{r}}^* - V_r^*\|_\infty \le \|\Delta r\|_\infty/(1-\gamma)$. □

**Remark 3.7.5** (Practical implications). Corollary 3.7.3 guarantees that **value iteration always converges**, regardless of initialization $V_0$. The rate (3.18) shows that after $k$ iterations, the error shrinks by $\gamma^k$. For $\gamma = 0.9$, we have $\gamma^{10} \approx 0.35$; for $\gamma = 0.99$, we need $k \approx 460$ iterations to reduce error by a factor of 100. This explains why high-discount RL is computationally expensive.

**Remark 3.7.6** (OPE preview — Direct Method). Off-policy evaluation (Chapter 9) can be performed via a **model-based Direct Method**: estimate $(\widehat{P}, \hat{r})$ and apply the policy Bellman operator repeatedly under the model to obtain

$$\widehat{V}^\pi := \lim_{k\to\infty} (\mathcal{T}^\pi_{\widehat{P},\hat{r}})^k V_0, \tag{3.22}$$

for any bounded $V_0$. The contraction property (with $\gamma < 1$ and bounded $\hat{r}$) guarantees existence and uniqueness of $\widehat{V}^\pi$. Chapter 9 develops full off-policy evaluation (IPS, DR, FQE), comparing the Direct Method previewed here to importance-weighted estimators.

**Remark 3.7.7** (The deadly triad — when contraction fails). The contraction property 3.7.1 guarantees convergence for **exact, tabular** value iteration. However, three ingredients common in deep RL can break this guarantee:

1. **Function approximation**: Representing $V$ or $Q$ via neural networks restricts us to a function class $\mathcal{F}$. The composed operator $\Pi_{\mathcal{F}} \circ \mathcal{T}$ (project-then-Bellman) is generally **not** a contraction.
2. **Bootstrapping**: TD methods update toward $r + \gamma V(s')$, using the current estimate $V$. Combined with function approximation, this can cause divergence.
3. **Off-policy learning**: Learning about one policy while following another introduces distribution mismatch.

The combination—function approximation + bootstrapping + off-policy—is Sutton's **deadly triad** ((Sutton and Barto 2018, sec. 11.3)). Classical counterexamples (e.g., Baird's) demonstrate that the resulting learning dynamics can diverge even with linear function approximation. Chapter 7 introduces target networks and experience replay as partial mitigations. The fundamental tension, however, remains unresolved in theory—deep RL succeeds empirically despite lacking the contraction guarantees we have established here. Understanding this gap between theory and practice is a central theme of Part III.

## 1.8 3.8 Connection to Contextual Bandits ($\gamma = 0$)

The **contextual bandit** from Chapter 1 is the special case $\gamma = 0$ (no state transitions, immediate rewards only).

**Definition 3.8.1** (Bandit Bellman Operator)

For a contextual bandit with Q-function $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ (the expected immediate reward from Chapter 1), the **bandit Bellman operator** is:

$$(\mathcal{T}_{\text{bandit}} V)(x) := \sup_{a \in \mathcal{A}} Q(x, a). \tag{3.19}$$

This is precisely the MDP Bellman operator (3.12) specialized to $\gamma = 0$, since in the bandit setting the one-step reward is $r(x, a) = Q(x, a)$ (and when $\mathcal{A}$ is finite the supremum is a maximum):

$$(\mathcal{T} V)(x) = \sup_a [r(x, a) + \gamma \cdot 0] = \sup_a Q(x, a) = (\mathcal{T}_{\text{bandit}} V)(x).$$

In particular, the right-hand side does not depend on $V$: bandits have no bootstrapping term, because there is no next-state value to propagate.

**Proposition 3.8.1** (Bandit operator fixed point in one iteration)

For bandits ($\gamma = 0$), the optimal value function is:

$$V^*(x) = \sup_{a \in \mathcal{A}} Q(x, a), \tag{3.20}$$

and value iteration converges in **one step**: $V_1 = \mathcal{T}_{\text{bandit}} V_0 = V^*$ for any $V_0$.

*Proof.* Since $\gamma = 0$, applying the Bellman operator:

$$(\mathcal{T}_{\text{bandit}} V)(x) = \sup_a Q(x, a) = V^*(x),$$

independent of $V$. Thus $V_1 = V^*$ for any $V_0$. □

**Remark 3.8.1** (Contrast with MDPs). For $\gamma > 0$, value iteration requires multiple steps because we must propagate value information backward through state transitions. For bandits, there are no state transitions—rewards are immediate—so the optimal value is the statewise supremum of the immediate $Q$-values. This is why Chapter 1 could focus on **learning** $Q(x, a)$ without explicitly constructing value functions.

**Remark 3.8.2** (Chapter 1 formulation). Recall from Chapter 1 the bandit optimality condition: the optimal value EQ-1.9 is attained by the greedy policy EQ-1.10, yielding

$$V^*(x) = \max_{a \in \mathcal{A}} Q(x, a), \quad Q(x, a) = \mathbb{E}_\omega[R(x, a, \omega)].$$

This is exactly (3.20). The bandit formulation is the $\gamma = 0$ MDP.

---

## 1.9 3.9 Computational Verification

We now implement value iteration and verify the convergence theory numerically.

### 1.9.1 3.9.1 Toy MDP: GridWorld Navigation

**Setup**: A $5 \times 5$ grid. Agent starts at $(0, 0)$, goal is $(4, 4)$. Actions: $\{\text{up}, \text{down}, \text{left}, \text{right}\}$. Rewards: $+10$ at goal, $-1$ per step (encourages shortest paths). Transitions: deterministic (move in chosen direction unless blocked by boundary).

```python
from __future__ import annotations

from dataclasses import dataclass
from typing import Tuple

import numpy as np


@dataclass
class GridWorldConfig:
    size: int = 5
    gamma: float = 0.9
    goal_reward: float = 10.0


class GridWorldMDP:
    """Deterministic GridWorld used in Section 3.9.1."""

    def __init__(self, cfg: GridWorldConfig | None = None) -> None:
        self.cfg = cfg or GridWorldConfig()
        self.size = self.cfg.size
        self.gamma = self.cfg.gamma
        self.goal_reward = self.cfg.goal_reward

        self.goal = (self.size - 1, self.size - 1)
        self.n_states = self.size * self.size
        self.n_actions = 4  # up, down, left, right

        self.P = np.zeros((self.n_states, self.n_actions, self.n_states))
        self.r = np.zeros((self.n_states, self.n_actions))

        for i in range(self.size):
            for j in range(self.size):
                s = self._state_index(i, j)
                if (i, j) == self.goal:
                    for a in range(self.n_actions):
                        self.P[s, a, s] = 1.0
                        self.r[s, a] = self.goal_reward
                    continue
                for a in range(self.n_actions):
                    i_next, j_next = self._next_state(i, j, a)
                    s_next = self._state_index(i_next, j_next)
                    self.P[s, a, s_next] = 1.0
                    self.r[s, a] = -1.0

    def _state_index(self, i: int, j: int) -> int:
        return i * self.size + j

    def _next_state(self, i: int, j: int, action: int) -> Tuple[int, int]:
        if action == 0:  # up
            return max(i - 1, 0), j
        if action == 1:  # down
            return min(i + 1, self.size - 1), j
        if action == 2:  # left
```

```
            return i, max(j - 1, 0)
        return i, min(j + 1, self.size - 1)  # right

    def bellman_operator(self, values: np.ndarray) -> np.ndarray:
        q_values = self.r + self.gamma * np.einsum("ijk,k->ij", self.P, values)
        return np.max(q_values, axis=1)

    def value_iteration(
        self,
        V_init: np.ndarray | None = None,
        *,
        max_iter: int = 256,
        tol: float = 1e-10,
    ) -> Tuple[np.ndarray, list[float]]:
        values = np.zeros(self.n_states) if V_init is None else V_init.copy()
        errors: list[float] = []
        for _ in range(max_iter):
            updated = self.bellman_operator(values)
            error = float(np.max(np.abs(updated - values)))
            errors.append(error)
            values = updated
            if error < tol:
                break
        return values, errors


def run_gridworld_convergence_check() -> None:
    mdp = GridWorldMDP()
    V_star, errors = mdp.value_iteration()

    start_state = mdp._state_index(0, 0)
    goal_state = mdp._state_index(*mdp.goal)
    expected_goal = mdp.goal_reward / (1.0 - mdp.gamma)

    print("iters", len(errors), "final_err", f"{errors[-1]:.3e}")
    print("V_start", f"{V_star[start_state]:.6f}")
    print("V_goal", f"{V_star[goal_state]:.6f}", "expected_goal", f"{expected_goal:.6f}")

    V_init = np.zeros(mdp.n_states)
    initial_gap = float(np.max(np.abs(mdp.bellman_operator(V_init) - V_init)))
    print("initial_gap", f"{initial_gap:.6f}")

    print("k  ||V_{k+1}-V_k||_inf  bound_from_EQ_3_18  bound_ok")
    for k, err in enumerate(errors[:10]):
        bound = (mdp.gamma**k / (1.0 - mdp.gamma)) * initial_gap
        bound_ok = err <= bound + 1e-9
        print(f"{k:2d}  {err:18.10f}  {bound:16.10f}  {bound_ok}")

    ratios = [
        errors[k] / errors[k - 1]
        for k in range(1, min(len(errors), 25))
        if errors[k - 1] > 1e-12
    ]
    tail = ratios[-8:]
    print("tail_ratios", " ".join(f"{r:.4f}" for r in tail))

    grid = V_star.reshape((mdp.size, mdp.size))
    print("grid")
    for i in range(mdp.size):
        print(" ".join(f"{grid[i, j]:7.2f}" for j in range(mdp.size)))


run_gridworld_convergence_check()
```

Output:

```
iters 242 final_err 9.386e-11
V_start 37.351393
V_goal 100.000000 expected_goal 100.000000
initial_gap 10.000000
k  ||V_{k+1}-V_k||_inf  bound_from_EQ_3_18  bound_ok
 0       10.0000000000    100.0000000000  True
 1        9.0000000000     90.0000000000  True
 2        8.1000000000     81.0000000000  True
 3        7.2900000000     72.9000000000  True
 4        6.5610000000     65.6100000000  True
 5        5.9049000000     59.0490000000  True
 6        5.3144100000     53.1441000000  True
 7        4.7829690000     47.8296900000  True
 8        4.3046721000     43.0467210000  True
 9        3.8742048900     38.7420489000  True
tail_ratios 0.9000 0.9000 0.9000 0.9000 0.9000 0.9000 0.9000 0.9000
grid
  37.35   42.61   48.46   54.95   62.17
  42.61   48.46   54.95   62.17   70.19
  48.46   54.95   62.17   70.19   79.10
  54.95   62.17   70.19   79.10   89.00
  62.17   70.19   79.10   89.00  100.00
```

**Code ↔ Lab (Contraction Verification)**

We verify 3.7.3 and the rate bound (3.18) using the value-iteration listing above. The repository also includes a regression test that mirrors this computation: `tests/ch03/test_value_iteration.py`. - Run: `.venv/bin/pytest -q tests/ch03/test_value_iteration.py`

### 1.9.2 3.9.2 Analysis

The numerical experiment confirms:

1. **Convergence**: value iteration converges within the configured iteration budget, with final update size below the tolerance.
2. **Rate bound check**: the printed quantity $\|V_{k+1} - V_k\|_\infty$ remains below the right-hand side of (3.18) in the displayed iterations, providing a numerical sanity check on the contraction-based rate.
3. **Exponential decay**: consecutive error ratios are essentially constant at $\gamma = 0.9$, matching the contraction mechanism.
4. **Goal-state semantics**: since the goal is absorbing with per-step reward `goal_reward`, we obtain $V^*(\text{goal}) = \text{goal\_reward}/(1 - \gamma)$.

**Key observations**:

- The theoretical bound is **tight**: observed errors track $\gamma^k$ behavior closely

- Higher $\gamma$ (closer to 1) implies slower convergence: for $\gamma = 0.99$, convergence requires on the order of hundreds of iterations in this GridWorld.
- Value iteration is **robust**: it converges for any initialization $V_0$ (here $V_0 \equiv 0$)

---

## 1.10 3.10 RL Bridges: Previewing Multi-Episode Dynamics

In Chapter 11 we extend the within-session MDP of this chapter to an inter-session (multi-episode) MDP. Chapter 1's contextual bandit formalism and the discounted MDP formalism of this chapter both treat a single session in isolation. In practice, many objectives are inter-session: actions taken today influence the probability of future sessions and the distribution of future states.

The multi-episode formulation introduces three concrete changes:

1. Inter-session state transitions: the state includes variables such as satisfaction, recency, and loyalty tier, and these evolve across sessions as functions of engagement signals (clicks, purchases) and exogenous factors (seasonality).
2. Retention (hazard) modeling: a probabilistic mechanism decides whether another session occurs, based on the current inter-session state.
3. Long-term value across sessions: the return sums rewards over sessions, not only within a single session.

The operator-theoretic content does not change: once inter-session dynamics are part of the transition kernel, Bellman operators remain contractions under discounting, and value iteration remains a fixed-point method. Conceptually, this clarifies reward design. Chapter 1's reward EQ-1.2 includes $\delta \cdot \text{CLICKS}$ as a proxy for long-run value; in Chapter 11 we encode engagement into the state dynamics through retention, so long-run effects are represented without relying on a separate proxy term.

**Code $\leftrightarrow$ Reward (MOD-zoosim.dynamics.reward)**

Chapter 1's single-step reward EQ-1.2 maps to configuration and aggregation code: - Weights and defaults: `zoosim/core/config.py:195` (`RewardConfig`) - Engagement weight guardrail (
*delta/*
*alpha* bound): `zoosim/dynamics/reward.py:56` These safeguards keep $\delta$ small and bounded in the MVP regime while we develop multi-episode value in Chapter 11.

**Code $\leftrightarrow$ Simulator (MOD-zoosim.multi_episode.session_env, MOD-zoosim.multi_episode.retention)**

Multi-episode transitions and retention are implemented in the simulator: - Inter-session MDP wrapper: `zoosim/multi_episode/session_env.py:79` (`MultiSessionEnv.step`) - Retention probability (logistic hazard): `zoosim/multi_episode/retention.py:22` (`return_probability`) - Retention config: `zoosim/core/config.py:208` (`RetentionConfig`), `zoosim/core/config.py:216` (`base_rate`), `zoosim/core/config.py:217` (`click_weight`), `zoosim/core/config.py:218` (`satisfaction_weight`) In this regime, engagement enters via state transitions, aligning with the long-run objective previewed by EQ-1.2-prime and Chapter 11.

---

## 1.11 3.11 Summary: What We Have Built

This chapter established the operator-theoretic foundations of reinforcement learning:

Stochastic processes (Section 3.2–3.3): - Filtrations $(\mathcal{F}_t)$ model information accumulation over time - Stopping times $\tau$ capture random termination (session abandonment, purchase events) - Adapted processes ensure causality (policies depend on history, not future)

Markov Decision Processes (Section 3.4): - Formal tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ with standard Borel assumptions - Value functions $V^\pi(s)$, $Q^\pi(s, a)$ as expected cumulative rewards - Bellman equations (3.7) and (3.10) as recursive characterizations

Contraction theory (Section 3.6–3.7): - Banach fixed-point theorem 3.6.2 guarantees existence, uniqueness, and exponential convergence - Bellman operator $\mathcal{T}$ is a $\gamma$-contraction in sup-norm 3.7.1 - Value iteration $V_{k+1} = \mathcal{T} V_k$ converges at rate $\gamma^k$ 3.7.3 - Caveat: Contraction fails with function approximation (deadly triad, Remark 3.7.7)

Connection to bandits (Section 3.8): - Contextual bandits are the $\gamma = 0$ special case (no state transitions) - Chapter 1's formulation ([EQ-1.8], EQ-1.9, and [EQ-1.10]) is recovered exactly

Numerical verification (Section 3.9): - GridWorld experiment confirms theoretical convergence rate (3.18) - Exponential decay $\gamma^k$ observed empirically

What comes next:

- **Chapter 4–5**: Build the simulator (`zoosim`) with catalog, users, queries, click models
- **Chapter 6**: Implement LinUCB and Thompson Sampling for discrete template bandits
- **Chapter 7**: Continuous action optimization via $Q(x, a)$ regression
- **Chapter 9**: Off-policy evaluation (OPE) using importance sampling
- **Chapter 10**: Production guardrails (CM2 floors,
  *Delta*
  *textRank*@$k$ stability) applying CMDP theory from Section 3.5
- **Chapter 11**: Multi-episode MDPs with retention dynamics

All later algorithms—TD-learning, Q-learning, policy gradients—use Bellman operators as their organizing object, but their convergence guarantees require additional assumptions and are established case-by-case in later chapters. In Chapter 3, the contraction property yields a complete convergence story for exact dynamic programming, and the fixed-point theorem tells us what value iteration converges to.

---

## 1.12 3.12 Exercises

**Exercise 3.1** (Stopping times) [15 min]

Let $(S_t)$ be a user satisfaction process with $S_t \in [0, 1]$. Which of the following are stopping times?

(a) $\tau_1 = \inf\{t : S_t < 0.3\}$ (first time satisfaction drops below 0.3)
(b) $\tau_2 = \sup\{t \leq T : S_t \geq 0.8\}$ (last time satisfaction exceeds 0.8 before horizon $T$)
(c) $\tau_3 = \min\{t : S_{t+1} < S_t\}$ (first time satisfaction decreases)

Justify the answers using 3.2.4.

**Exercise 3.2** (Bellman equation verification) [15 min]

Consider a 2-state MDP with $\mathcal{S} = \{s_1, s_2\}$, $\mathcal{A} = \{a_1, a_2\}$, $\gamma = 0.9$. Transitions and rewards:

$$P(\cdot|s_1, a_1) = (0.8, 0.2), \quad r(s_1, a_1) = 5 \tag{5}$$
$$P(\cdot|s_1, a_2) = (0.2, 0.8), \quad r(s_1, a_2) = 10 \tag{6}$$
$$P(\cdot|s_2, a_1) = (0.5, 0.5), \quad r(s_2, a_1) = 2 \tag{7}$$
$$P(\cdot|s_2, a_2) = (0.3, 0.7), \quad r(s_2, a_2) = 8 \tag{8}$$

Given $V(s_1) = 50$, $V(s_2) = 60$, compute $(\mathcal{T}V)(s_1)$ and $(\mathcal{T}V)(s_2)$ using (3.12).

**Exercise 3.3** (Contraction property) [20 min]

Prove that the Bellman expectation operator $\mathcal{T}^\pi$ for a fixed policy $\pi$ (defined in [EQ-3.9]) is a $\gamma$-contraction, using a similar argument to 3.7.1.

**Exercise 3.4** (Value iteration implementation) [extended: 30 min]

Implement value iteration for the GridWorld MDP from Section 3.9.1, but with **stochastic transitions**: with probability 0.8, the agent moves in the intended direction; with probability 0.2, it moves in a random perpendicular direction. Verify that:

(a) Value iteration still converges
(b) The convergence rate satisfies (3.18)
(c) The optimal policy changes (compare to deterministic case)

### 1.12.1 Labs

- Lab 3.1 — Contraction Ratio Tracker: execute the GridWorld contraction experiment and compare empirical ratios against the $\gamma$ bound in (3.16).
- Lab 3.2 — Value Iteration Wall-Clock Profiling: sweep multiple discounts, log iteration counts, and tie the scaling back to 3.7.3 (value iteration convergence rate).

**Exercise 3.5** (Bandit special case) [10 min]

Verify that for $\gamma = 0$, the Bellman operator (3.12) reduces to the bandit operator (3.19). Explain why value iteration converges in one step for bandits.

**Exercise 3.6** (Discount factor exploration) [20 min]

Using the GridWorld code from Section 3.9.1, run value iteration for $\gamma \in \{0.5, 0.7, 0.9, 0.99\}$. Plot the number of iterations required for convergence (tolerance $10^{-6}$) as a function of $\gamma$. Explain the relationship using (3.18).

**Exercise 3.7** (RL preview: Policy evaluation) [extended: 30 min]

Implement **policy evaluation** (iterative computation of $V^\pi$ for a fixed policy $\pi$ using [EQ-3.8]). For the GridWorld MDP:

(a) Define a suboptimal policy $\pi$: always go right unless at right edge (then go down)
(b) Compute $V^\pi$ via policy evaluation: $V_{k+1} = \mathcal{T}^\pi V_k$
(c) Compare $V^\pi$ to $V^*$ (from value iteration)
(d) Verify that $V^\pi(s) \leq V^*(s)$ for all $s$ (why must this hold?)

---

## 1.13 References

See `docs/references.bib` for full citations.

Key references for this chapter: - (Puterman 2014) — Definitive MDP textbook (Puterman) - (Bertsekas 2012) — Dynamic programming and optimal control (Bertsekas) - (Folland 1999) — Measure theory and functional analysis foundations - (Brezis 2011) — Banach space theory and operator methods - (Sutton and Barto 2018) — Modern RL textbook and the deadly triad discussion

---

## 1.14 3.13 Production Checklist

**Production Checklist (Chapter 3)**

- **Seeds**: Ensure RNGs for stochastic MDPs use fixed seeds from `SimulatorConfig.seed` for reproducibility - **Discount factor**: Document $\gamma$ choice in config files; highlight $\gamma \to 1$ convergence slowdown - **Numerical stability**: Use double precision (`float64`) for value iteration to avoid accumulation errors - **Cross-references**: Update Knowledge Graph (`docs/knowledge_graph/graph.yaml`) with all theorem/definition IDs - **Tests**: Add regression tests for value iteration convergence (verify

(3.18) bounds programmatically)

---

## 1.15 Exercises & Labs

Companion material for Chapter 3 lives in:

- Exercises and runnable lab prompts: `docs/book/ch03/exercises_labs.md`
- Worked solutions with printed outputs: `docs/book/ch03/ch03_lab_solutions.md`

Reproducibility checks:

- Chapter 3 regression test: `.venv/bin/pytest -q tests/ch03/test_value_iteration.py`
- Run all Chapter 3 labs: `.venv/bin/python scripts/ch03/lab_solutions.py --all`

Bertsekas, Dimitri P. 2012. *Dynamic Programming and Optimal Control*. 4th ed. Vol. 1. Athena Scientific.

Brezis, Haïm. 2011. *Functional Analysis, Sobolev Spaces and Partial Differential Equations*. Universitext. Springer.

Folland, Gerald B. 1999. *Real Analysis: Modern Techniques and Their Applications*. 2nd ed. Wiley.

Puterman, Martin L. 2014. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. John Wiley & Sons.

Sutton, Richard S., and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*. 2nd ed. MIT Press.