Asynchronous Coagent Networks: Stochastic Networks for Reinforcement Learning without Backpropagation or a Clock

James Kostas, Chris Nota, and Philip S. Thomas
College of Information and Computer Sciences
University of Massachusetts Amherst
{jekostas,cnota,pthomas}@cs.umass.edu

Abstract

In this paper we introduce a reinforcement learning (RL) approach for training policies, including artificial neural network policies, that is both *backpropagation-free* and *clock-free*. It is *backpropagation-free* in that it does not propagate any information backwards through the network. It is *clock-free* in that no signal is given to each node in the network to specify when it should compute its output and when it should update its weights. We contend that these two properties increase the biological plausibility of our algorithms and facilitate distributed implementations. Additionally, our approach eliminates the need for customized learning rules for hierarchical RL algorithms like the option-critic.

1 Introduction

Reinforcement learning (RL) algorithms share qualitative similarities with the algorithms implemented by animal brains. However, there remain clear differences between these two types of algorithms. For example, while RL algorithms using artificial neural networks require information to flow backwards through the network via the *backpropagation algorithm*, there is currently debate about whether this is feasible in biological neural implementations (Werbos and Davis, 2016). *Policy gradient coagent networks* (PGCNs) are a class of RL algorithms that were introduced to remove this possibly biologically implausible property of RL algorithms—they use artificial neural networks but do not use the backpropagation algorithm (Thomas, 2011).

Since their introduction, PGCN algorithms have proven to be not only a possible improvement in biological plausibility, but a practical tool for improving RL agents. They were used to solve RL problems with high-dimensional action spaces (Thomas and Barto, 2012), are the RL precursor to the more general *stochastic computation graphs* (Schulman et al., 2015), and, as we will show in this paper, generalize the recently proposed *option-critic* architecture (Bacon et al., 2017), while drastically simplifying key derivations.

The paper introducing PGCNs claims that each node (neuron) in a network can perform all of its updates given only local information—information that would be available to a neuron in an animal brain. However, this is not the case since PGCNs still require an implicit signal that was overlooked: a clock to determine when each node should produce its output and update its weights. In this paper we show how PGCNs can be extended to operate without a clock signal (or with a noisy clock signal), resulting in a new class of RL algorithms that **1)** do not require the

backpropagation of information through an artificial neural network, and **2)** do not require a clock signal to be broadcast to any nodes in the network. Furthermore, removing the need for a clock has important ramifications beyond biological plausibility: It allows distributed implementations of large neural networks to operate without requirements of synchronicity, and provides an alternate view of temporal abstraction for RL algorithms. We clarify this second point later by discussing the relationship between PGCN algorithms and the options framework (Sutton et al., 1999).

The contributions of this paper are: **1)** a complete and formal proof of a key result related to PGCN algorithms that this paper relies on, and which prior work provides an informal and incomplete proof, **2)** a generalization of the PGCN framework to handle asynchronous networks, **3)** a proof that asynchronous PGCNs generalize the option-critic framework, and **4)** empirical support of our theoretical claims regarding the gradients of asynchronous PGCN algorithms.

2 Related Work

Klopf (1982) theorized that traditional models of classical and operant conditioning could be explained by modeling biological neurons as *hedonistic*, that is, seeking excitation and avoiding inhibition. The ideas motivating coagent networks bear a deep resemblance to Klopf's proposal.

Stochastic neural networks were first applied to RL at the dawn of machine learning itself, with applications dating back at least to Marvin Minsky's *stochastic neural analog reinforcement calculator* (SNARC), built in 1951 Russell and Norvig (2016). Interest in their usage has continued throughout the history of RL. The well-known REINFORCE algorithm was originally proposed with the intent of training stochastic networks Williams (1992), though it has since been primarily applied to conventional networks. Other rules like *adaptive reward-penalty* Barto (1985) were proposed exclusively for training stochastic networks.

Multi-agent reinforcement learning (MARL) is the application of RL in *multi-agent systems*. MARL differs from coagent RL in that agents typically have separate manifestations within the environment; additionally, the goals of the agents may or may not be aligned. Despite these differences, many results from the study of MARL are relevant to the study of coagent networks. For instance, Liu et al. (2014) showed that multi-agent systems sometimes learn more quickly when agents are given individualized rewards, rather than only receiving team-wide rewards. An overview of MARL is given by Busoniu et al. (2010).

Deep reinforcement learning, the application of conventional neural networks to RL, has recently become an active area of research, following its successful application to challenging domains such as real-time games Mnih et al. (2015), board games Silver et al. (2017), and robotic control Andrychowicz et al. (2018). While conventional deep networks have dominated recent RL research (and machine learning research more broadly), stochastic networks have also recently been a moderately popular research topic. The formalism of *stochastic computation graphs* was proposed to describe networks with a mixture of stochastic and deterministic nodes, with applications to supervised learning, unsupervised learning, and RL Schulman et al. (2015). *Policy gradient coagent networks* Thomas (2011), the subject of this paper, were proposed for RL specifically, and have been used to discover "motor primitives" in simulated robotic control tasks Thomas and Barto (2012).

Several recently proposed approaches fit into the formalism of stochastic networks, but the relationship has frequently gone unnoticed. One notable example is the *option-critic* architecture Bacon et al. (2017). The option-critic provides a framework for learning *options* Sutton et al. (1999), a type of high-level and temporally extended action, and how to choose between options. The motivations for the option-critic are largely similar to previous motivations for PGCNs: namely,

achieving temporal abstraction and hierarchical control. We show that the option-critic architecture can be described by a particular coagent network architecture. The theory presented in this paper makes the resulting derivation of the option-critic policy gradients nearly trivial in contrast with the original derivations, and places the option-critic within a more general theoretical framework.

3 Background

We consider an MDP, $M = (\mathcal{S}, \mathcal{A}, \mathcal{R}, P, R, d_0, \gamma)$, where $\mathcal{S}$ is the finite set of possible *states*, $\mathcal{A}$ is the finite set of possible *actions*, and $\mathcal{R}$ is the finite set of possible *rewards*. Let $t \in \{0, 1, 2, \dots\}$ denote the time step. S_t, A_t, and R_t are the state, action, and reward at time t, and are random variables that take values in $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{R}$, respectively. $P{:}\mathcal{S}\times\mathcal{A}\times\mathcal{S}\to[0,1]$ is the *transition function*, given by $P(s, a, s') \coloneqq \Pr(S_{t+1}{=}s' | S_t{=}s, A_t{=}a)$. $R{:}\mathcal{S}\times\mathcal{A}\times\mathcal{S}\times\mathcal{R}\to[0,1]$ is the *reward distribution*, given by $R(s, a, s', r) \coloneqq \Pr(R_t{=}r | S_t{=}s, A_t{=}a, S_{t+1}{=}s')$. The *initial state distribution*, $d_0{:}\mathcal{S} \to [0, 1]$, is given by $d_0(s) \coloneqq \Pr(S_0{=}s)$. The *discount factor*, $\gamma \in [0, 1]$, is the reward discount parameter. An *episode* is a sequence of states, actions, and rewards, starting from $t{=}0$ and continuing indefinitely. We assume that the discounted sum of rewards over an episode is finite.

A policy, $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, is a stochastic method of selecting actions, such that $\pi(s, a) \coloneqq \Pr(A_t{=}a | S_t{=}s)$. A *parameterized policy* is a policy that takes a parameter vector $\theta \in \mathbb{R}^n$. Different parameter vectors result in different policies. More formally, we redefine the symbol π to denote a parameterized policy, $\pi : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^n \to [0, 1]$, such that for all $\theta \in \mathbb{R}^n$, $\pi(\cdot, \cdot, \theta)$ is a policy. We assume that $\partial\pi(s, a, \theta)/\partial\theta$ exists for all $s \in \mathcal{S}, a \in \mathcal{A}, \theta \in \mathbb{R}^n$. An agent's goal is typically to choose a policy that maximizes the *objective function*, which is defined as $J(\pi) \coloneqq \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t | \pi\right]$, where conditioning on π denotes that, for all t, $A_t \sim \pi(S_t, \cdot)$. The state-value function, $v^\pi : \mathcal{S} \to \mathbb{R}$, is defined as $v^\pi(s) \coloneqq \mathbf{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k} \middle| S_t{=}s, \pi\right]$. The discounted return, G_t, is defined as $G_t \coloneqq \sum_{k=0}^{\infty} \gamma^k R_{t+k}$. We denote the objective function for a policy that has parameters θ as $J(\theta)$, and condition probabilities on θ to denote that the parameterized policy uses parameter vector θ.

Consider a parameterized policy that consists of an acyclic network of nodes, called *coagents*, which do not share parameters. Each coagent can have several inputs that may include the state at time t, a noisy and incomplete observation of the state at time t, and/or the outputs of other coagents. When considering the i^{th} coagent, θ can be partitioned into two vectors, $\theta_i \in \mathbb{R}^{n_i}$ (the parameters used by the i^{th} coagent) and $\bar{\theta}_i \in \mathbb{R}^{n-n_i}$ (the parameters used by all other coagents). From the point of view of the i^{th} coagent, A_t is produced from S_t in three stages: execution of the nodes prior to the i^{th} coagent (nodes whose outputs are required to compute the input to the i^{th} coagent), execution of the i^{th} coagent, and execution of the remaining nodes in the network to produce the final action. This process is depicted graphically in Figure 1 and described in detail below. First, we define a parameterized distribution $\pi_i^{\text{pre}}(S_t, \cdot, \bar{\theta}_i)$ to capture how the previous coagents in the network produce their outputs given the current state. The output of the previous coagents is a random variable, which we denote by U_t^{pre}, and which takes continuous and/or discrete values in some set $\mathcal{U}^{\text{pre}}$. U_t^{pre} is sampled from the distribution $\pi_i^{\text{pre}}(S_t, \cdot, \bar{\theta}_i)$. Next, the i^{th} coagent takes S_t and U_t^{pre} as input and produces the output U_t^i (below, when it is unambiguously referring to the output of the i^{th} coagent, we make the i implicit and denote it as U_t). We denote this input, (S_t, U_t^{pre}), as X_t (or X_t^i if it is not unambiguously referring to the i^{th} coagent). The conditional distribution of U_t^i is given by the i^{th} coagent's policy, $\pi_i(X_t, \cdot, \theta_i)$. Although we allow the i^{th} coagent's output to depend directly on S_t, it may be parameterized to only depend on U_t^{pre}. Finally, A_t is sampled according to a distribution $\pi_i^{\text{post}}(X_t, U_t^i, \cdot, \bar{\theta}_i)$, which captures how the subsequent coagents in the network produce A_t. Below, we sometimes make $\bar{\theta}_i$ and θ_i implicit and write the three policy functions as

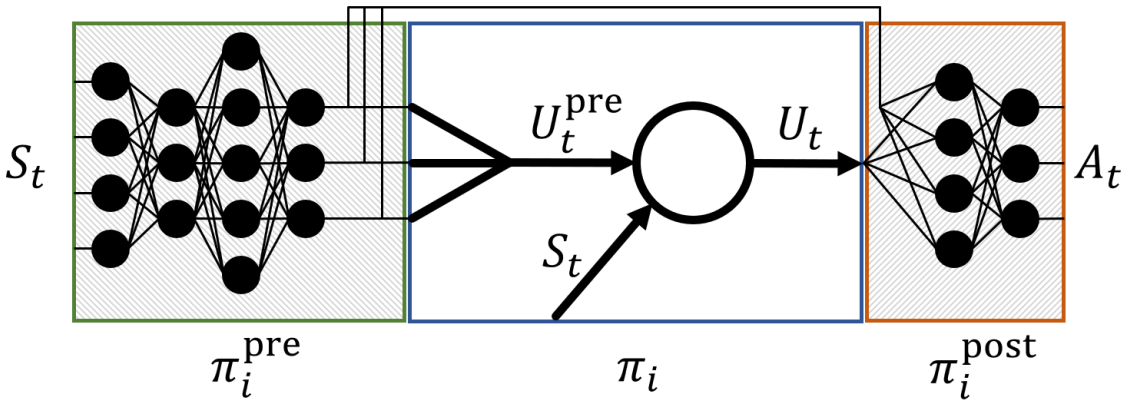

Figure 1: Example diagram of three step process for action generation, for a fully connected feedforward network (we do not require the network to have this structure). The circle in the middle denotes the i^{th} coagent. In the first step, preceding nodes are executed to compute the inputs to this coagent. In the second step the coagent uses these inputs to produce its output, U_t. In the third step the remainder of the network is executed to produce an action.

$\pi_i^{\text{pre}}(S_t, \cdot)$, $\pi_i(X_t, \cdot)$, and $\pi_i^{\text{post}}(X_t, U_t^i, \cdot)$. Also, following the work of Thomas and Barto (2011), we model the i^{th} coagent's environment (consisting of the original environment as well as all other coagents in the network) as an MDP called a *conjugate Markov decision process* (CoMDP).

4 The Coagent Policy Gradient Theorem

Consider what would happen if the i^{th} coagent ignored all of the complexity in this problem setup and simply implemented an unbiased policy gradient algorithm, like REINFORCE (Williams, 1992). From the i^{th} coagent's point of view, the state would be S_t and U_t^{pre} together (the coagent may ignore components of this state, such as the $\mathcal{S}$ component), its actions would be U_t, and the rewards would remain R_t. We refer to the expected update in this setting as the *local policy gradient*, Δ_i, for the i^{th} coagent. Note that although we eventually prove that, for all θ_i, $\Delta_i(\theta_i)$ is equivalent to the policy gradient of the i^{th} CoMDP, we do not assume this equivalence. Formally, the local policy gradient of the i^{th} coagent is:

$$\Delta_i(\theta_i) \coloneqq \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t G_t \frac{\partial \ln\left(\pi_i\left(X_t, U_t, \theta_i\right)\right)}{\partial \theta_i} \middle| \theta\right].$$

What would happen if all coagents updated their parameters using a local and unbiased policy gradient algorithm? The *coagent policy gradient theorem* (CPGT) answers this question: If θ is fixed and all coagents update their parameters following unbiased estimates, $\widehat{\Delta}_i(\theta_i)$, of their local policy gradients, then the entire network will follow an unbiased estimator of $\nabla J(\theta)$, which we call the *global policy gradient*. For example, if every coagent performs the following update simultaneously at the end of each episode, then the entire network will be performing stochastic gradient ascent on

J (without using backpropagation):

$$\theta_i \leftarrow \theta_i + \alpha \sum_{t=0}^{\infty} \gamma^t G_t \left(\frac{\partial \ln \left(\pi_i \left(X_t, U_t, \theta_i \right) \right)}{\partial \theta_i} \right).$$

In practice, one would use a more sophisticated policy gradient algorithm than this simple variant of REINFORCE.

Although Thomas and Barto (2011) present the CPGT in their Theorem 3, the provided proof is lacking in two ways. First, it is not general enough for our purposes because it only considers networks with two coagents. Second, it is missing a crucial step. They define a new MDP, the CoMDP, which models the environment faced by a coagent. They show that the policy gradient for this new MDP is a component of $\nabla J(\theta)$. However, they do not show that the chosen definition of the CoMDP accurately describes the environment that the coagent faces. Without this step, Thomas and Barto (2011) have shown that there is a new MDP for which the policy gradient is a component of $\nabla J(\theta)$, but not that this MDP has any relation to the coagent network. In this section we provide formal and generalized proofs of the CPGT. Although this proof is an important contribution of this work, due to space restrictions this section is an abbreviated outline of the proof in Section A of the supplementary material.

4.1 Conjugate Markov Decision Process (CoMDP)

We model the i^{th} coagent's environment as an MDP, called the CoMDP, and begin by formally defining the i^{th} CoMDP. Given M, i, π_i^{pre}, π_i^{post}, and $\bar{\theta}_i$, we define a corresponding CoMDP, M^i, as $M^i \coloneqq (\mathcal{X}^i, \mathcal{U}^i, \mathcal{R}^i, P^i, R^i, d_0^i, \gamma_i)$, where:

- We write $\tilde{X}_t^i$, $\tilde{U}_t^i$, and $\tilde{R}_t^i$ to denote the state, action, and reward of M^i at time t. Below, we relate these random variables to the corresponding random variables in M. Note that all *random variables* in the CoMDP are written with tildes to provide a visual distinction between terms from the CoMDP and original MDP. Additionally, when it is clear that we are referring to the i^{th} CoMDP, we often make i implicit and denote these as $\tilde{X}_t$, $\tilde{U}_t$, and $\tilde{R}_t$.
- $\mathcal{X}^i \coloneqq \mathcal{S} \times \mathcal{U}_i^{\text{pre}}$. We often denote $\mathcal{X}^i$ simply as $\mathcal{X}$. This is the input (analogous to a state set) to the i^{th} coagent. Additionally, for $x \in \mathcal{X}$, we denote the $\mathcal{S}$ component as $x.s$ and the $\mathcal{U}^{\text{pre}}$ component as $x.u_{\text{pre}}$. We also sometimes denote an $x \in \mathcal{X}^i$ as $(x.s, x.u_{\text{pre}})$. For example, $\Pr(\tilde{X}_t^i{=}(s, u_{\text{pre}}))$ represents the probability that $\tilde{X}_t$ has $\mathcal{S}$ component s and $\mathcal{U}^{\text{pre}}$ component u_{pre}.
- $\mathcal{U}^i$ (or simply $\mathcal{U}$) is an arbitrary set that denotes the output of the i^{th} coagent.
- $\mathcal{R}^i \coloneqq \mathcal{R}$ and $\gamma_i \coloneqq \gamma$.
- $\forall x \in \mathcal{X}\ \forall x' \in \mathcal{X}\ \forall u \in \mathcal{U}\ \forall \bar{\theta}_i \in \mathbb{R}^{n-n_i}$,

$$P^i(x, u, x', \bar{\theta}_i) \coloneqq \pi_i^{\text{pre}}(x'.s, x'.u_{\text{pre}}) \sum_{a \in \mathcal{A}} P(x.s, a, x'.s) \pi_i^{\text{post}}(x, u, a),$$

  Below, we make $\bar{\theta}_i$ implicit and denote this as $P^i(x, u, x')$. Recall from the definition of an MDP and its relation to the transition function that this means: $P^i(x, u, x') = \Pr(\tilde{X}_{t+1}{=}x' | \tilde{X}_t{=}x, \tilde{U}_t{=}u)$.

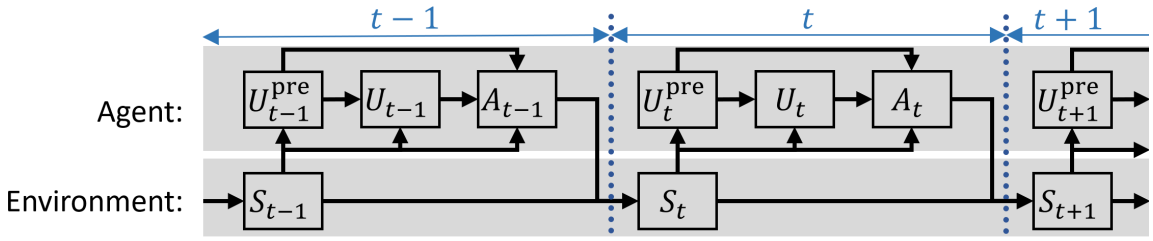

Figure 2: Bayesian network depicting the relationships of relevant random variables. Independence properties can be established by d-separation. Note that these causal properties only apply to the MDP M discussed above; any such properties of the CoMDPs will be explicitly proven.

- $\forall x \in \mathcal{X}\ \forall x' \in \mathcal{X}\ \forall u \in \mathcal{U}\ \forall r \in \mathcal{R}^i\ \forall \bar{\theta}_i \in \mathbb{R}^{n-n_i}$,

$$R^i(x, u, x', r, \bar{\theta}_i) \coloneqq \sum_{a \in \mathcal{A}} R(x.s, a, x'.s, r) \frac{P(x.s, a, x'.s)\pi_i^{\text{post}}(x, u, a)}{\sum_{\hat{a} \in \mathcal{A}} P(x.s, \hat{a}, x'.s)\pi_i^{\text{post}}(x, u, \hat{a})}.$$

  Like the transition function, we make $\bar{\theta}_i$ implicit and write $R^i(x, u, x', r)$.

- $\forall x \in \mathcal{X},\ d_0^i(x) \coloneqq d_0(x.s)\pi_i^{\text{pre}}(x.s, x.u_{\text{pre}})$.

We write $J_i(\theta_i)$ to denote the objective function of M^i. Notice that although $\bar{\theta}_i$ (the parameters of the other coagents) is not an explicit parameter of the objective function, it is implicitly included via the CoMDP's transition function.

We assume that, given the same parameters θ_i, the i^{th} coagent has the same policy in both the original MDP and the i^{th} CoMDP. That is,

Assumption 1.

$$\forall s \in \mathcal{S},\ \forall u_{\text{pre}} \in \mathcal{U}^{\text{pre}},\ \forall u \in \mathcal{U},\ \forall \theta_i \in \mathbb{R}^i,\ \pi_i((s, u_{\text{pre}}), u, \theta_i) = \Pr(\tilde{U}_t{=}u | \tilde{X}_t{=}(s, u_{\text{pre}}), \theta_i).$$

4.2 The CoMDP Models the Coagent's Environment

Here we show that our definition of the CoMDP correctly models the coagent's environment. We do so by presenting a series of properties and lemmas that each establish different components of the relationship between the CoMDP and the environment faced by a coagent. Figure 2 depicts the setup that we have described and makes relevant independence properties clear. The proofs of these properties and theorems are provided in the supplementary material.

In Properties 1 and 2, by manipulating the definitions of d_0^i and π_i^{pre}, we show that d_0^i and the distribution of $\tilde{X}_0.s$ capture the distribution of the inputs to the i^{th} coagent.

Property 1. $\forall x \in \mathcal{X}, d_0^i(x) = \Pr(S_0{=}x.s, U_0^{\text{pre}}{=}x.u_{\text{pre}})$.

Property 2. *For all* $s \in \mathcal{S}$, $\Pr(\tilde{X}_0.s{=}s) = d_0(s)$.

In Property 3, we show that P^i captures the distributions of the inputs that the i^{th} coagent will see given the input at the previous step and the output that it selected.

Property 3. *For all $x \in \mathcal{X}$, $x' \in \mathcal{X}$, and $u \in \mathcal{U}$,*

$$P^i(x, u, x') = \Pr(S_{t+1}{=}x'.s, U^{\text{pre}}_{t+1}{=}x'.u_{\text{pre}} | S_t{=}x.s, U^{\text{pre}}_t{=}x.u_{\text{pre}}, U_t{=}u).$$

In Property 4, we show that R^i captures the distribution of the rewards that the i^{th} coagent receives given the output that it selected and the inputs at the current and next steps.

Property 4. *For all $x \in \mathcal{X}$, $x' \in \mathcal{X}$, $u \in \mathcal{U}$, and $r \in \mathcal{R}$,*

$$R^i(x, u, x', r) = \Pr(R_t{=}r | S_t{=}x.s, U^{\text{pre}}_t{=}x.u_{\text{pre}}, U_t{=}u, S_{t+1}{=}x'.s, U^{\text{pre}}_{t+1}{=}x'.u_{\text{pre}}).$$

In Properties 5 and 6, we show that the distributions of $\tilde{X}$ and $\tilde{X}_t.s$ capture the distribution of inputs to the i^{th} coagent.

Property 5. *For all $s{\in}\mathcal{S}$ and $u_{\text{pre}}{\in}\mathcal{U}^{\text{pre}}_i$,*

$$\Pr(\tilde{X}_t{=}(s, u_{\text{pre}})) = \Pr(S_t{=}s, U^{\text{pre}}_t{=}u_{\text{pre}}).$$

Property 6. *For all $s \in \mathcal{S}$,*

$$\Pr(\tilde{X}_t.s{=}s) = \Pr(S_t{=}s).$$

In Property 7, we show that the distribution of $\tilde{X}_t.u_{pre}$ given $\tilde{X}_t.s$ captures the distribution π^{pre}_i.

Property 7. *For all $s \in \mathcal{S}$ and $u_{\text{pre}} \in \mathcal{U}^{\text{pre}}_i$,*

$$\Pr(\tilde{X}_t.u_{\text{pre}}{=}u_{\text{pre}} | \tilde{X}_t.s{=}s) = \pi^{\text{pre}}_i(s, u_{\text{pre}}).$$

In Property 8, we show that the distribution of $\tilde{X}_{t+1}.s$ given $\tilde{X}_t.s$, $\tilde{X}_t.u_{pre}$, and $\tilde{U}_t$ captures the distribution of the $\mathcal{S}$ component of the input that the i^{th} coagent will see given the input at the previous step and the output that it selected.

Property 8. *For all $s \in \mathcal{S}$, $s' \in \mathcal{S}$, $u_{\text{pre}} \in \mathcal{U}^{\text{pre}}_i$, and $u \in \mathcal{U}$,*

$$\begin{aligned}&\Pr(\tilde{X}_{t+1}.s{=}s' | \tilde{X}_t.s{=}s, \tilde{X}_t.u_{\text{pre}}{=}u_{\text{pre}}, \tilde{U}_t{=}u)\\ =&\Pr(S_{t+1}{=}s' | S_t{=}s, U^{\text{pre}}_t{=}u_{\text{pre}}, U_t{=}u).\end{aligned}$$

In Property 9, we use Property 8 to show that: Given the $\mathcal{S}$ component of the input, the $\mathcal{U}^{\text{pre}}_i$ component of the input that the i^{th} coagent will see is independent of the previous input and output.

Property 9. *For all $s \in \mathcal{S}$, $s' \in \mathcal{S}$, $u_{\text{pre}} \in \mathcal{U}^{\text{pre}}_i$, $u'_{\text{pre}} \in \mathcal{U}^{\text{pre}}_i$, and $u \in \mathcal{U}$,*

$$\begin{aligned}&\Pr(\tilde{X}_{t+1}.u_{\text{pre}}{=}u'_{\text{pre}} | \tilde{X}_{t+1}.s{=}s')\\ =&\Pr(\tilde{X}_{t+1}.u_{\text{pre}}{=}u'_{\text{pre}} | \tilde{X}_{t+1}.s{=}s', \tilde{X}_t{=}(s, u_{\text{pre}}), \tilde{U}_t{=}u).\end{aligned}$$

In Property 10, we use Assumption 1 and Properties 6, 7, 8, 9, and 10 to show that the distribution of $\tilde{R}^i_t$ captures the distribution of the rewards that the i^{th} coagent receives.

Property 10. *For all* $r \in \mathcal{R}$, $\Pr(R_t{=}r) = \Pr(\tilde{R}^i_t{=}r)$.

We then use Properties 3 and 4 and the definition of M^i to show that:

Lemma 1. *M^i is a Markov decision process.*

Finally, in Lemma 2, we use the properties above to show that the CoMDP M^i (built from M, i, π_i^{pre}, π_i^{post}, and $\bar{\theta}_i$) correctly models the local environment of the i^{th} coagent.

Lemma 2. *For all $M, i, \pi_i^{\text{pre}}, \pi_i^{\text{post}}$, and $\bar{\theta}_i$, and given a policy parameterized by θ_i, the corresponding CoMDP M^i satisfies Properties 1-6 and Property 10.*

Lemma 2 is stated more formally in the supplementary material.

4.3 The Coagent Policy Gradient Theorem

We use Property 10 to show that, given the same θ, the objective functions produce the same output in the original MDP and all CoMDPs. More formally:

Property 11. *For all coagents i, for any θ, $J(\theta){=}J_i(\theta_i)$.*

Next, using Lemmas 1 and 2, we show that the local policy gradient, Δ_i (the expected value of the naive REINFORCE update), is equivalent to the gradient $\frac{\partial J_i}{\partial \theta_i}$ of the i^{th} CoMDP.

Lemma 3. *For all coagents i, for all θ_i, $\frac{\partial J_i(\theta_i)}{\partial \theta_i} = \Delta_i(\theta_i)$.*

We can now formally state and prove the CPGT. Using Property 11 and Lemma 3, we show that the local policy gradients are the components of the global policy gradient:

Theorem 1 (Coagent Policy Gradient Theorem)**.**
$\nabla J(\theta) = [\Delta_1(\theta_1)^\intercal, \Delta_2(\theta_2)^\intercal, \ldots, \Delta_m(\theta_m)^\intercal]^\intercal$*, where m is the number of coagents and Δ_i is the local policy gradient of the i^{th} coagent.*

Corollary 1. *If α_t is a deterministic positive stepsize, $\sum_{t=0}^{\infty} \alpha_t = \infty$, $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$, additional technical assumptions are met (Bertsekas and Tsitsiklis, 2000, Proposition 3), and each coagent updates its parameters, θ_i, with an unbiased local policy gradient update $\theta_i \leftarrow \theta_i + \alpha_t \widehat{\Delta}_i(\theta_i)$, then $J(\theta)$ converges to a finite value and $\lim_{t\to\infty} \nabla J(\theta){=}0$.*

5 Asynchronous Recurrent Networks

Having formally established the CPGT, we now turn to extending the PGCN framework to asynchronous and cyclic networks—networks where the coagents *execute*, that is, look at their local state and choose actions, asynchronously and without any necessary order. This extension removes the necessity for a biologically implausible *perfect* clock, allowing the network to function with an imprecise clock, or with no clock at all. This also allows for distributed implementations, where nodes may not execute synchronously.

We first consider how we may modify an MDP to allow coagents to execute at arbitrary points in time, including at points *in between* our usual time steps. We make a simplifying assumption: Time is discrete (as opposed to continuous). We break a time step of the MDP into an arbitrarily large number of shorter steps, which we call *atomic time steps*. We assume that the environment

performs its usual update regularly every $n \in \mathbb{Z}^+$ atomic time steps, and that each coagent *executes* (chooses an output in its respective $\mathcal{U}^i$) at each atomic time step with some probability, given by an arbitrary but fixed probability distribution. The duration of atomic time steps can be arbitrarily small to allow for arbitrarily close approximations to continuous time or to model, for example, a CPU cluster that performs billions of updates per second. The objective is still the expected value of G_0, the discounted sum of rewards from all atomic time steps: $J(\theta) = \mathbf{E}[G_0|\theta] = \mathbf{E}[\sum_{t=0}^{\infty} \gamma^t R_t|\theta]$.

Next, we extend the coagent framework to allow *cyclic* connections. Previously, we considered a coagent's local state to be captured by $X_t^i = (S_t, U_t^{\text{pre}})$, where U_t^{pre} is some combination of outputs from coagents that come before the i^{th} coagent topologically. We now allow coagents to also consider the output of all m coagents on the *previous* time step, $U_{t-1}^{\text{all}} = \{U_{t-1}^1, U_{t-1}^2, \dots U_{t-1}^m\}$. In the new setting, the local state at time t is therefore given by $X_t^i = (S_t, U_t^{\text{pre}}, U_{t-1}^{\text{all}})$. The corresponding local state set is given by $\mathcal{X}^i = \mathcal{S} \times \mathcal{U}^{\text{pre}} \times \mathcal{U}^1 \times \cdots \times \mathcal{U}^m$. In this construction, when $t = 0$, we must consider some initial output of each coagent, U_{-1}^{all}. For the i^{th} coagent, we define U_{-1}^i to be drawn from some independent initial distribution, h_0^i, such that for all $u \in \mathcal{U}^i, h_0^i(u) = \Pr(U_{-1}^i = u)$.

We redefine how each coagent selects actions in the asynchronous setting. First, we define a random variable, E_t^i, the value of which is 1 if the i^{th} coagent executes on atomic time step t, and 0 otherwise. One useful factor for deciding whether a coagent should update or not is the number of time steps since its last execution. To this end, we define a *counter* for each coagent given at time t for the i^{th} coagent by $C_t^i \in \mathbb{N}$ (define $\mathbb{N}$ to be the set of non-negative integers). The counter increments by one at each atomic time step, and resets to zero when the coagent executes. We define a function, f_c, that captures this behavior, such that $C_{t+1} = f_c(C_t, E_t)$. Each coagent has a fixed *execution function*, $\beta_i : \mathcal{X}^i \times \mathbb{N} \to [0, 1]$, which defines the probability of the i^{th} coagent executing on time step t, given the coagent's local state and its counter. That is, for all $x \in \mathcal{X}^i$ and $c \in \mathbb{N}$, $\beta_i(x, c) \coloneqq \Pr(E_t^i = 1 | X_t^i = x, C_t^i = c)$. Finally, the action, U_t^i, that the i^{th} coagent selects at time t is sampled from $\pi_i(X_t^i, \cdot, \theta_i)$ if $E_t^i = 1$, and is U_{t-1}^i otherwise. That is, if the agent does not execute on atomic time step t, then it should repeat its action from time $t - 1$.

This *asynchronous setting* does not match the usual MDP description because the policy represented by the network is *non-Markovian*—that is, we cannot determine the distribution over the output of the network given only the current state, S_t. Therefore, we cannot apply the CPGT. However, we show that the asynchronous setting can be reduced to the usual acyclic, synchronous setting using formulaic changes to the state set, transition function, and network structure. This allows us to derive an expression for the gradient with respect to the parameters of the original, asynchronous network, and therefore to train such a network. We prove a result similar to the CPGT that allows us to update the parameters of each coagent using only states and actions from atomic time steps when the coagent executes.

5.1 The CPGT for Asynchronous Networks

We first extend the definition of the *local policy gradient*, Δ_i, to the asynchronous setting. In the synchronous setting, the local policy gradient captures the update that a coagent would perform if it was following an unbiased policy gradient algorithm using its local inputs and outputs. In the asynchronous setting, we capture the update that an agent would perform if it were to consider only the local inputs and outputs it sees when it executes. Formally, we define the *asynchronous local policy gradient*:

$$\Delta_i(\theta_i) \coloneqq \mathbf{E}\left[\sum_{t=0}^{\infty} E_t^i \gamma^t G_t \frac{\partial \ln\left(\pi_i\big((S_t, U_t^{\text{pre}}), U_t, \theta_i\big)\right)}{\partial \theta_i} \middle| \theta\right].$$

The only change from the synchronous version is the introduction of E_t^i. Note that when the coagent does not execute ($E_t^i = 0$), the entire inner expression is 0. In other words, these states and actions can be ignored. An algorithm estimating G_t would still need to consider the rewards from *every* atomic time step, including time steps where the coagent does not execute. However, the algorithm may still be designed such that the coagents only perform a computation when executing. For example, during execution, coagents may be given the discounted sum of rewards since their last execution. The important question is then: does something like the CPGT hold for the *asynchronous* local policy gradient? If each coagent executes a policy gradient algorithm using unbiased estimates of Δ_i, does the network still perform gradient descent on the asynchronous setting objective, J? The answer turns out to be yes.

Theorem 2 (Asynchronous Coagent Policy Gradient Theorem)**.**

$$\nabla J(\theta) = [\Delta_1(\theta_1)^\intercal, \Delta_2(\theta_2)^\intercal, \ldots, \Delta_m(\theta_m)^\intercal]^\intercal \,,$$

where m is the number of coagents and Δ_i is the asynchronous local policy gradient of the i^{th} coagent.

Proof. The general approach is to show that for any MDP M, with an asynchronous network represented by π with parameters θ, there is an *augmented* MDP, $\grave{M}$, with objective $\grave{J}$ and an *acylic, synchronous* network, $\grave{\pi}$, with the same parameters θ, such that $J(\theta) = \grave{J}(\theta)$. Thus, we *reduce* the asynchronous problem to an equivalent synchronous problem. Applying the CPGT to this *reduced setting* allows us to derive Theorem 2.

The original MDP, M, is given by the tuple $(\mathcal{S}, \mathcal{A}, P, R, d_0, \gamma)$. We define the *augmented* MDP, $\grave{M}$, as the tuple, $(\grave{\mathcal{S}}, \grave{\mathcal{A}}, \grave{P}, \grave{R}, \grave{d}_0, \grave{\gamma})$. We would like $\grave{M}$ to hold *all* of the information necessary for each coagent to compute its next output, including the previous outputs and counters of all coagents. This will allow us to construct an acyclic version of the network to which we may apply the CPGT. We define $\mathcal{U}^{\text{all}} = \{\mathcal{U}^1, \mathcal{U}^2, \ldots, \mathcal{U}^m\}$ to be the combined output set of all m coagents in π, $\mathcal{C} = \mathbb{N}^m$ to be the set of possible counter values, and $\mathcal{E} = \{0, 1\}^m$ to be the set of possible combinations of coagent executions. We define the state set to be $\grave{\mathcal{S}} = \mathcal{S} \times \mathcal{U}^{\text{all}} \times \mathcal{C}$, and the action set to be $\grave{\mathcal{A}} = \mathcal{A} \times \mathcal{U}^{\text{all}} \times \mathcal{E}$. We write the random variables representing the state, action, and reward at time t as $\grave{S}_t$, $\grave{A}_t$, and $\grave{R}_t$ respectively. Additionally, we refer to the components of values $\grave{s} \in \grave{\mathcal{S}}$ and $\grave{a} \in \grave{\mathcal{A}}$ and the components of the random variables $\grave{S}_t$ and $\grave{A}_t$ using the same notation as for the components of X_t above (for example, $\grave{s}.s$ is the $\mathcal{S}$ component of $\grave{s}$, $\grave{A}_t.u^{\text{all}}$ is the $\mathcal{U}^{\text{all}}$ component of $\grave{A}_t$, etc.). For vector components, we write the i^{th} component of the vector using a subscript i (for example, $\grave{s}.u_i^{\text{all}}$ is the i^{th} component of $\grave{s}.u^{\text{all}}$).

The transition function, $\grave{P}$, captures the original transition function, the fact that $\grave{S}_{t+1}.u^{\text{all}} = \grave{A}_t.u^{\text{all}}$, and the behavior of the counters. For all $\grave{s}, \grave{s}' \in \grave{S}$ and $\grave{a} \in \grave{A}$, $\grave{P}(\grave{s}, \grave{a}, \grave{s}')$ is given by $P(\grave{s}.s, \grave{a}.a, \grave{s}'.s)$ if $\grave{s}'.u^{\text{all}}{=}\grave{a}.u^{\text{all}}$ and $\grave{s}'.c{=}f_c(\grave{s}.c, \grave{a}.e)$, and 0 otherwise. For all $\grave{s}, \grave{s}' \in \grave{S}, \grave{a} \in \grave{A}$, and $r \in \mathbb{R}$, the reward distribution is simply given by $\grave{R}(\grave{s}, \grave{a}, \grave{s}', r){=}R(\grave{s}.s, \grave{a}.a, \grave{s}'.s, r)$. The initial state distribution, $\grave{d}_0$, captures the original state distribution, the initialization of each coagent, and the initialization of the counters to zero. For all $\grave{s}{\in}\grave{S}$, it is given by $\grave{d}_0(\grave{s}){=}d_0(\grave{s}.s)\prod_{i=1}^m h_0(\grave{s}.u_i)$ if $\grave{s}.c$ is the zero vector, and 0 otherwise. The discount parameter is $\grave{\gamma}{=}\gamma$. The objective is the usual: $\grave{J}(\theta){=}\mathbf{E}[\grave{G}_0|\theta]$, where $\grave{G}_0{=}\mathbf{E}[\sum_{t=0}^{\infty}\grave{\gamma}^t\grave{R}_t|\theta]$.

We next define the synchronous network, $\grave{\pi}$, in terms of components of the original asynchronous network, π—specifically, each π_i, β_i, and θ_i. We must modify the original network to accept inputs in $\grave{\mathcal{S}}$ and produce outputs in $\grave{\mathcal{A}}$. Recall that in the asynchronous network, the local state at time t of the i^{th} coagent is given by $X_t^i = (S_t, U_t^{\text{pre}}, U_{t-1}^{\text{all}})$. In the augmented MDP, the information in U_{t-1}^{all} is

contained in $\grave{S}_t$, so the local state of the i^{th} coagent in the synchronous network is $\grave{X}_t^i = (\grave{S}_t, \grave{U}_t^{\text{pre}})$, with accompanying state set $\grave{\mathcal{X}}^i = \grave{\mathcal{S}} \times \grave{\mathcal{U}}^{\text{pre}}$. To produce the $\mathcal{U}^{\text{all}}$ component of the action, $\grave{A}_t.u^{\text{all}}$, we append the output of each coagent to the action. In doing so, we have removed the need for cyclic connections, but still must deal with the asynchronous execution.

The critical step is as follows: We represent each coagent in the asynchronous network by *two* coagents in the the synchronous network, the first of which represents the execution function, β_i, and the second of which represents the original policy, π_i. At time step t, the first coagent accepts $\grave{X}_t^i$ and outputs 1 with probability $\beta_i((\grave{S}_t.s, \grave{U}_t^{\text{pre}}, \grave{S}_t.u), \grave{S}_t.c_i)$, and 0 otherwise. We append the output of every such coagent to the action in order to produce the $\mathcal{E}$ component of the action, $\grave{A}_t.e$. Because the coagent representing β_i executes before the coagent representing π_i, from the latter's perspective, the output of the former is present in $\grave{U}_t^{\text{pre}}$, that is, $\grave{U}_t^{\text{pre}}.e_i = \grave{A}_t.e_i$. If $\grave{U}_t^{\text{pre}}.e_i = 1$, the coagent samples a new action from π_i. Otherwise, it repeats its previous action, which can be read from its local state (that is, $\grave{X}_t^i.u_i^{\text{all}} = \grave{U}_{t-1}^i$). Formally, for all $(\grave{s}, \grave{u}_{\text{pre}}) \in \grave{\mathcal{X}}^i$ and θ_i, the probability of the latter coagent producing action $\grave{u} \in \grave{\mathcal{U}}^i$ is given by:

$$\grave{\pi}_i((\grave{s}, \grave{u}_{\text{pre}}), \grave{u}, \theta_i) := \begin{cases} \pi_i((\grave{s}.s, \grave{u}_{\text{pre}}, \grave{s}.u^{\text{all}}), \grave{u}, \theta_i), & \text{if } \grave{u}_{\text{pre}}.e_i{=}1 \\ 1, & \text{if } \grave{u}_{\text{pre}}.e_i{=}0 \text{ and } \grave{s}.u_i^{\text{all}}{=}\grave{u} \\ 0, & \text{otherwise.} \end{cases}$$

This completes the description of $\grave{\pi}$. In the supplementary material, we prove that this network exactly captures the behavior of the asynchronous network—that is, $\grave{\pi}((s,u,c),(a,u',e),\theta) = \Pr(A_t = a, U_t^{\text{all}} = u', E_t = e | S_t = s, U_{t-1}^{\text{all}} = u, C_t = c, \theta)$ for all possible values of $a, u, c, a, u', e,$ and θ in their appropriate sets.

The proof that $J(\theta) = \grave{J}(\theta)$ is given in Section B of the supplementary material, but it follows intuitively from the fact that 1) the "hidden" state of the network is now captured by the state set, 2) $\grave{\pi}$ accurately captures the dynamics of the hidden state, and 3) this hidden state does not materially affect the transition function or the reward distribution with respect to the original states and actions.

Having shown that the expected return in the asynchronous setting is equal to the expected return in the synchronous setting, we turn to deriving the asynchronous local policy gradient, Δ_i. It follows from $J(\theta) = \grave{J}(\theta)$ that $\nabla J(\theta) = \nabla \grave{J}(\theta)$. Since $\grave{\pi}$ is a synchronous, acylic network, and $\grave{M}$ is an MDP, we can apply the CPGT to find an expression for $\nabla \grave{J}(\theta)$. This gives us for the i^{th} coagent in the synchronous network:

$$\frac{\partial \grave{J}(\theta)}{\partial \theta_i} = \mathbf{E}\left[\sum_{t=0}^{\infty} \grave{\gamma}^t \grave{G}_t \frac{\partial \ln \left(\grave{\pi}_i \left((\grave{S}_t, \grave{U}_t^{\text{pre}}), \grave{U}_t, \theta_i \right) \right)}{\partial \theta_i} \middle| \theta \right].$$

Consider $\partial \ln \left(\grave{\pi}_i((\grave{S}_t, \grave{U}_t^{\text{pre}}), \grave{U}_t, \theta_i) \right) / \partial \theta_i$, which we abbreviate as $\partial \grave{\pi}_i / \partial \theta_i$. When $\grave{U}_t^{\text{pre}}.e_i = 0$, we know that the action is $\grave{U}_t^i = \grave{S}_t.u_i = \grave{U}_{t-1}^i$ regardless of θ. Therefore, in these local states, $\partial \grave{\pi}_i / \partial \theta_i$ is zero. When $\grave{U}_t^{\text{pre}}.e_i = 1$, we see from the definition of $\grave{\pi}$ that $\partial \grave{\pi}_i / \partial \theta_i {=} \partial \pi_i / \partial \theta_i$. Therefore, we see that the in all cases, $\partial \grave{\pi}_i / \partial \theta_i {=} (\grave{U}_t^{\text{pre}}.e_i) \partial \pi / \partial \theta_i$. Substituting this into the above expression yields:

$$\mathbf{E}\left[\sum_{t=0}^{\infty} (\grave{U}_t^{\text{pre}}.e_i) \grave{\gamma}^t \grave{G}_k \frac{\partial \ln \left(\pi_i \left((\grave{S}_t.s, \grave{U}_t^{\text{pre}}, S_t.u^{\text{all}}), \grave{U}_t, \theta_i \right) \right)}{\partial \theta_i} \middle| \theta \right].$$

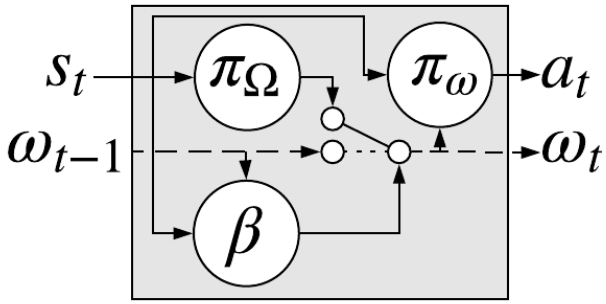

Figure 3: The option-critic framework Bacon et al. (2017) depicted as a coagent network. The network receives the previous state, s_t, and option, ω_{t-1}. The termination function, β, chooses whether to continue the previous option, or to allow π_Ω to select a new option. This choice is depicted by the position of the switch in the center of the figure. π_ω chooses a new action, a_t, based on the resulting state-option pair.

In the proof that $J(\theta) = \grave{J}(\theta)$ given in Section B of the supplementary material, we show that the distribution over all over the analogous random variables is equivalent in both settings (for example, for all $s \in \mathcal{S}$, $\Pr(S_t = s) = \Pr(\grave{S}_t.s = s)$). Substituting each of the random variables of M into the above expression yields precisely the asynchronous local policy gradient, Δ_i. □

6 Case Study: Option-Critic

The *option-critic* framework Bacon et al. (2017) aspires to many of the same goals as coagent networks: namely, hierarchical learning and temporal abstraction. In this section, we show that the architecture is equivalently described in terms of a simple, three-node coagent network, depicted in Figure 3. We show that the policy gradients derived by Bacon et al. (2017) are equivalent to the gradients suggested by the CPGT. While the original derivation required several steps, we show that the derivation is nearly trivial using the CPGT, demonstrating its utility as a tool for analyzing a variety of important RL algorithms.

In this section, we adhere mostly to the notation given by Bacon et al. (2017), with some minor changes used to enhance conceptual clarity regarding the inputs and outputs of each policy. In the option-critic framework, the agent is given a set of *options*, Ω. The agent selects an option, $\omega \in \Omega$, by sampling from a policy $\pi_\Omega : \mathcal{S} \times \Omega \to [0, 1]$. An action, $a \in \mathcal{A}$, is then selected from a policy which considers both the state and the current option: $\pi_\omega : (\mathcal{S} \times \Omega) \times \mathcal{A} \to [0, 1]$. A new option is not selected at every time step; rather, an option is run until a *termination function*, $\beta : (\mathcal{S} \times \Omega) \times \{0, 1\} \to [0, 1]$, selects the termination action, 0. If the action 1 is selected, then the current option continues. π_ω is parameterized by weights θ, and β by weights ϑ. Bacon et al. (2017)

gave the corresponding policy gradients, which we rewrite as:

$$\frac{\partial J}{\partial \theta} = \sum_{x \in (\mathcal{S} \times \Omega)} d^{\pi}_{\Omega}(x) \sum_{a \in \mathcal{A}} \frac{\partial \pi_{\omega}(x, a)}{\partial \theta} Q_U(x, a), \tag{1}$$

$$\frac{\partial J}{\partial \vartheta} = - \sum_{x \in (\mathcal{S} \times \Omega)} d^{\pi}_{\Omega}(x) \frac{\partial \beta(x, 0)}{\partial \vartheta} A_{\Omega}(x.s, x.\omega), \tag{2}$$

where $d^{\pi}_{\Omega}(x)$ is a discounted weighting of state-option pairs, given by $d^{\pi}_{\Omega}(x) \coloneqq \sum_{t=0}^{\infty} \gamma^t \Pr(s_t = x.s, \omega_t{=}x.\omega)$, $Q_U(x, a)$ is the expected return from choosing option $x.\omega$ and action a at state s under the current policy, and $A_{\Omega}(s, \omega)$ is the advantage of choosing option ω, given by $A_{\Omega}(s, \omega) = Q_{\Omega}(s, \omega) - V_{\Omega}(s)$, where $Q_{\Omega}(s, \omega)$ is the expected return from choosing option ω in state s, and $V_{\Omega}(s)$ is the expected return from beginning in state s with no option selected. No policy gradient theorem is provided for π_{Ω}, but Bacon et al. (2017) suggest policy gradient methods at the SMDP level, planning, or *intra-option Q-learning* Sutton et al. (1999). Another option is to use the asynchronous local policy gradient presented in Section 5. This has the benefit of providing a unified approach to training the network, rather than the piecemeal suggestions above.

6.1 PGCN Equivalence

Previously, the CPGT was written in terms of expected values. An equivalent expression of the local gradient for policy π_i is the sum over the local state set, $\mathcal{X}_i$, and the local action set, $\mathcal{U}_i$: $\partial J(\theta)/\partial \theta_i = \sum_{x \in \mathcal{X}_i} d^{\pi}_i(x) \sum_{u \in \mathcal{U}_i} \frac{\partial \pi_i(x,u)}{\partial \theta_i} Q_i(x, u)$, where $Q_i(x, u) = \mathbf{E}[G_t | X^i_t{=}x, U^i_t{=}u]$. Deriving the policy gradient for a particular coagent simply requires identifying the inputs and outputs and plugging them into this formula. A network implementing the option critic in the general case is shown in Figure 3. We show that applying the PCGT to this network yields (1) and (2).

First consider the policy gradient for π_{ω}, that is, $\partial J/\partial \theta$: the input set is $X_{\omega} = \mathcal{S} \times \Omega$, and the action set is $\mathcal{A}$. The local initial state distribution (the d^{π}_i term) is given exactly by d^{π}_{Ω}, and the local state-action value function (the Q_i term) is given exactly by Q_U. Directly substituting these terms into the CPGT immediately yields (1). Note that this derivation is completely trivial using the CPGT: only direct substitution is required. In contrast, the original derivation from Bacon et al. (2017) required a degree of complexity and several steps.

Next consider $\partial J/\partial \vartheta$. The input set is again $X_{\beta} = \mathcal{S} \times \Omega$, but the action set is $\{0, 1\}$. The local state distribution is again the distribution over state-option pairs, given by d^{π}_{Ω}. The PCGN expression gives us

$$\frac{\partial J}{\partial \vartheta} = \sum_{x \in (\mathcal{S} \times \Omega)} d^{\pi}_{\Omega}(x) \sum_{u \in \{0,1\}} \frac{\partial \beta(x, u)}{\partial \vartheta} Q_{\beta}(x, u).$$

We will show that this is equivalent to (2). Note that we only have two actions, whose probabilities must sum to one. Therefore, the gradients of the policy are equal in magnitude but opposite in sign. That is, for all $x \in (\mathcal{S} \times \Omega)$: $\beta(x, 0) + \beta(x, 1) = 1$, so $\frac{\partial \beta(x,0)}{\partial \vartheta} = -\frac{\partial \beta(x,1)}{\partial \vartheta}$. Additionally, we know that $Q_{\beta}(x, 1)$ is the expected value of continuing option $x.\omega$ in state $x.s$, given by $Q_{\Omega}(x.s, x.\omega)$. $Q_{\beta}(x, 0)$ is the expected value of choosing a new action in state $x.s$, given by $V_{\Omega}(x.s)$, and therefore,

$Q_\beta(x,1) - Q_\beta(x,0) = A_\Omega(x.s, x.\omega)$. The full derivation is:

$$\begin{aligned}
\frac{\partial J}{\partial \vartheta} &= \sum_{x \in (\mathcal{S} \times \Omega)} d^\pi_\Omega(x) \left[\frac{\partial \beta(x,0)}{\partial \vartheta} Q_\beta(x,0) + \frac{\partial \beta(x,1)}{\partial \vartheta} Q_\beta(x,1) \right] \\
&= - \sum_{x \in (\mathcal{S} \times \Omega)} d^\pi_\Omega(x) \frac{\partial \beta(x,0)}{\partial \vartheta} (Q_\beta(x,1) - Q_\beta(x,0)) \\
&= - \sum_{x \in (\mathcal{S} \times \Omega)} d^\pi_\Omega(x) \frac{\partial \beta(x,0)}{\partial \vartheta} A_\Omega(x.s, x.\omega).
\end{aligned}$$

We see that the result is exactly equivalent to (2). While this derivation required some minor symbol manipulation, the CPGT greatly simplified the derivation to the point of near triviality relative to the approach used by Bacon et al. (2017). In fact, it was not strictly necessary to reduce the gradient to this form: It is reasonable to implement an algorithm using the raw result of the CPGT. Using the option-critic framework as an example, we have shown that the CPGT is an enormously useful tool for simplifying the derivation of the policy gradients for arbitrary stochastic networks.

7 Empirical Support for the CPGT

To empirically support the CPGT, we compare the gradient (∇J) estimates of the CPGT and a brute force method. Finite difference methods are a well-established technique for computing the gradient of a function from samples; they serve as a straightforward baseline to evaluate the gradients produced by our algorithm. We expect these estimates to approach the same value as the amount of data used approaches infinity. For the purposes of testing the CPGT, we use a simple toy problem described in Section C of the supplementary material. The results are presented in Figure 4; this data provides empirical support for the CPGT. The coagents asynchronously execute; the environment updates every step and each coagent has a 0.5 probability of executing each step. The gradient estimates appear to converge, providing empirical support for the CPGT.

8 Conclusion

We provide a formal and general proof of the coagent policy gradient theorem (CPGT) for stochastic policy networks, and extend it to the asynchronous and recurrent setting. This result demonstrates that, if coagents apply standard policy gradient algorithms from the perspective of their inputs and outputs, then the entire network will follow the policy gradient, even in asynchronous or recurrent settings, or those without a clock. We empirically support the CPGT, and show that the option-critic framework is a special case of the CPGT. Future work will focus on the potential for massive parallelization of asynchronous coagent networks, and on the potential for many levels of implicit temporal abstraction through varying coagent execution rates.

References

Paul J Werbos and Joshua JJ Davis. Regular cycles of forward and backward signal propagation in prefrontal cortex and in consciousness. *Frontiers in Systems Neuroscience*, 10:97, 2016.

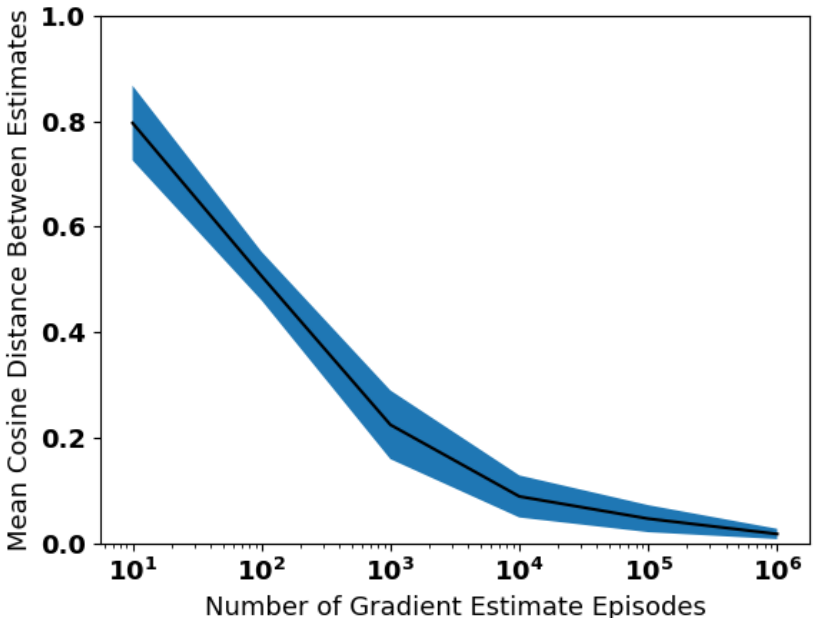

Figure 4: The average (across coagents) cosine distance between the gradient estimates of the finite difference method and the CPGT (vertical axis) versus the number of episodes used for the CPGT estimate (horizontal axis). This data is drawn from 20 trials. Error bars represent standard error. 5×10^8 episodes are used for each finite difference estimate. As the amount of data used for the CPGT gradient estimate increases, the cosine distance approaches zero, indicating that the two gradient estimates converge to the same value as the amount of data increases.

P. S. Thomas. Policy gradient coagent networks. In *Advances in Neural Information Processing Systems 24*, pages 1944–1952. 2011.

P. S. Thomas and A. G. Barto. Motor primitive discovery. In *Procedings of the IEEE Conference on Development and Learning and Epigenetic Robotics*, pages 1–8, 2012.

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. In *Advances in Neural Information Processing Systems*, pages 3528–3536, 2015.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *AAAI*, pages 1726–1734, 2017.

Richard S Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.

A Harry Klopf. *The hedonistic neuron: A theory of memory, learning, and intelligence*. Toxicology-Sci, 1982.

Stuart J Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Malaysia; Pearson Education Limited,, 2016.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

Andrew G Barto. Learning by statistical cooperation of self-interested neuron-like computing elements. 1985.

Bingyao Liu, Satinder Singh, Richard L Lewis, and Shiyin Qin. Optimal rewards for cooperative agents. *IEEE Transactions on Autonomous Mental Development*, 6(4):286–297, 2014.

Lucian Busoniu, Robert Babuška, and Bart De Schutter. Multi-agent reinforcement learning: An overview. 2010.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354, 2017.

Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*, 2018.

P. S. Thomas and A. G. Barto. Conjugate Markov decision processes. In *Proceedings of the Twenty-Eighth International Conference on Machine Learning*, pages 137–144, 2011.

D. P. Bertsekas and J. N. Tsitsiklis. Gradient convergence in gradient methods. *SIAM J. Optim.*, 10:627–642, 2000.

Supplementary Material

A Complete CPGT Proofs

Assumption 1. $\forall s \in \mathcal{S}\ \forall u_{\text{pre}} \in \mathcal{U}^{\text{pre}}\ \forall u \in \mathcal{U}\ \forall \theta_i \in \mathbb{R}^i,\ \pi_i((s, u_{\text{pre}}), u, \theta_i) = \Pr(\tilde{U}_t = u | \tilde{X}_t = (s, u_{\text{pre}}), \theta_i)$.

Property 1.

$$\forall x \in \mathcal{X},\ d_0^i(x) = \Pr(S_0 = x.s, U_0^{\text{pre}} = x.u_{\text{pre}}).$$

Proof.

$$\begin{aligned} d_0^i(x) =& d_0(x.s)\pi_i^{\text{pre}}(x.s, x.u_{\text{pre}}) \\ \overset{(a)}{=}& \Pr(S_0 = x.s) \Pr(U_0^{\text{pre}} = x.u_{\text{pre}} | S_0 = x.s) \\ =& \Pr(S_0 = x.s, U_0^{\text{pre}} = x.u_{\text{pre}}), \end{aligned}$$

where **(a)** follows from the definitions of π_i^{pre} and d_0. □

Property 2.

$$\forall s \in \mathcal{S},\ \Pr(\tilde{X}_0.s = s) = d_0(s).$$

Proof.

$$\begin{aligned} \Pr(\tilde{X}_0.s = s) \overset{(a)}{=}& \sum_{u_{\text{pre}} \in \mathcal{U}^{\text{pre}}} \Pr(\tilde{X}_0.s = s, \tilde{X}_0.u_{\text{pre}} = u_{\text{pre}}) \\ \overset{(b)}{=}& \sum_{u_{\text{pre}} \in \mathcal{U}^{\text{pre}}} d_0^i((s, u_{\text{pre}})) \\ \overset{(c)}{=}& \sum_{u_{\text{pre}} \in \mathcal{U}^{\text{pre}}} d_0(s)\pi_i^{\text{pre}}(s, u_{\text{pre}}) \\ =& d_0(s) \underbrace{\sum_{u_{\text{pre}} \in \mathcal{U}^{\text{pre}}} \Pr(U_t^{\text{pre}} = u_{\text{pre}} | S_t = s)}_{=1} \\ =& d_0(s), \end{aligned}$$

where **(a)** follows from marginalization over u_{pre}, **(b)** follows from the definition of the initial state distribution for an MDP, and **(c)** follows from the definition of d_0^i for the CoMDP (see Property 1). □

Property 3.

$$\forall x \in \mathcal{X}\ \forall x' \in \mathcal{X}\ \forall u \in \mathcal{U},\ P^i(x, u, x') = \Pr(S_{t+1} = x'.s, U_{t+1}^{\text{pre}} = x'.u_{\text{pre}} | S_t = x.s, U_t^{\text{pre}} = x.u_{\text{pre}}, U_t = u).$$

Proof.

$$\begin{aligned} P^i(x, u, x') =& \pi_i^{\text{pre}}(x'.s, x'.u_{\text{pre}}) \sum_{a \in \mathcal{A}} P(x.s, a, x'.s)\pi_i^{\text{post}}(x, u, a) \\ \overset{(a)}{=}& \sum_{a \in \mathcal{A}} \pi_i^{\text{post}}(x, u, a) \Pr(S_{t+1} = x'.s | S_t = x.s, A_t = a) \Pr(U_{t+1}^{\text{pre}} = x'.u_{\text{pre}} | S_{t+1} = x'.s) \\ \overset{(b)}{=}& \sum_{a \in \mathcal{A}} \pi_i^{\text{post}}(x, u, a) \Pr(S_{t+1} = x'.s | S_t = x.s, A_t = a) \\ & \times \Pr(U_{t+1}^{\text{pre}} = x'.u_{\text{pre}} | S_{t+1} = x'.s, S_t = x.s, A_t = a), \end{aligned}$$

where **(a)** follows from the definitions of π_i^{pre} and the transition function P and **(b)** follows from M's conditional independence properties. The definition of conditional probability allows us to combine the last two terms:

$$
\begin{aligned}
P^i(x,u,x') &= \sum_{a\in\mathcal{A}} \pi_i^{\text{post}}(x,u,a)\Pr(S_{t+1}=x'.s, U_{t+1}^{\text{pre}}=x'.u_{\text{pre}}|S_t=x.s, A_t=a)\\
&\overset{\text{(a)}}{=} \sum_{a\in\mathcal{A}} \Pr(A_t=a|S_t=x.s, U_t^{\text{pre}}=x.u_{\text{pre}}, U_t=u)\\
&\quad\times\Pr(S_{t+1}=x'.s, U_{t+1}^{\text{pre}}=x'.u_{\text{pre}}|S_t=x.s, U_t^{\text{pre}}=x.u_{\text{pre}}, U_t=u, A_t=a)\\
&\overset{\text{(b)}}{=} \Pr(S_{t+1}=x'.s, U_{t+1}^{\text{pre}}=x'.u_{\text{pre}}|S_t=x.s, U_t^{\text{pre}}=x.u_{\text{pre}}, U_t=u),
\end{aligned}
$$

where **(a)** follows from the definition of π_i^{post} and the application of M's independence properties and **(b)** follows from marginalization over a. □

Property 4.

$$
\begin{aligned}
&\forall x\in\mathcal{X}\ \forall x'\in\mathcal{X}\ \forall u\in\mathcal{U}\ \forall r\in\mathcal{R},\ R^i(x,u,x',r)\\
=&\Pr(R_t=r|S_t=x.s, U_t^{\text{pre}}=x.u_{\text{pre}}, U_t=u, S_{t+1}=x'.s, U_{t+1}^{\text{pre}}=x'.u_{\text{pre}}).
\end{aligned}
$$

Proof.

$$
\begin{aligned}
R^i(x,u,x',r) &\coloneqq \sum_{a\in\mathcal{A}} R(x.s,a,x'.s,r)\frac{P(x.s,a,x'.s)\pi_i^{\text{post}}(x,u,a)}{\sum_{\hat{a}\in\mathcal{A}}P(x.s,\hat{a},x'.s)\pi_i^{\text{post}}(x,u,\hat{a})}\\
&\overset{\text{(a)}}{=} \sum_{a\in\mathcal{A}} R(x.s,a,x'.s,r)P(x.s,a,x'.s)\pi_i^{\text{post}}(x,u,a)\\
&\quad\div\Bigg[\sum_{\hat{a}\in\mathcal{A}}\Pr(S_{t+1}=x'.s|S_t=x.s, A_t=\hat{a}, U_t^{\text{pre}}=x.u_{\text{pre}}, U_t=u)\\
&\quad\times\Pr(A_t=\hat{a}|S_t=x.s, U_t^{\text{pre}}=x.u_{\text{pre}}, U_t=u)\Bigg]\\
&\overset{\text{(b)}}{=} \frac{\sum_{a\in\mathcal{A}} R(x.s,a,x'.s,r)P(x.s,a,x'.s)\pi_i^{\text{post}}(x,u,a)}{\Pr(S_{t+1}=x'.s|S_t=x.s, U_t^{\text{pre}}=x.u_{\text{pre}}, U_t=u)},
\end{aligned}
$$

where **(a)** follows from the definitions of terms in the denominator and M's conditional independence properties (applied to the first term in the denominator) and **(b)** follows from marginalization over $\hat{a}$. Expanding the definitions of the remaining terms, we get:

$$
\begin{aligned}
R^i(x,u,x',r) =&\frac{\sum_{a\in\mathcal{A}}\Pr(R_t=r|S_t=x.s, A_t=a, S_{t+1}=x'.s)\Pr(S_{t+1}=x'.s|S_t=x.s, A_t=a)}{\Pr(S_{t+1}=x'.s|S_t=x.s, U_t^{\text{pre}}=x.u_{\text{pre}}, U_t=u)}\\
&\times\Pr(A_t=a|S_t=x.s, U_t^{\text{pre}}=x.u_{\text{pre}}, U_t=u)\\
\overset{\text{(a)}}{=}&\frac{1}{\Pr(S_{t+1}=x'.s|S_t=x.s, U_t^{\text{pre}}=x.u_{\text{pre}}, U_t=u)}\\
&\times\sum_{a\in\mathcal{A}}\Pr(R_t=r|S_t=x.s, A_t=a, S_{t+1}=x'.s, U_t^{\text{pre}}=x.u_{\text{pre}}, U_t=u)\\
&\times\Pr(S_{t+1}=x'.s|S_t=x.s, A_t=a, U_t^{\text{pre}}=x.u_{\text{pre}}, U_t=u)\\
&\times\Pr(A_t=a|S_t=x.s, U_t^{\text{pre}}=x.u_{\text{pre}}, U_t=u),
\end{aligned}
$$

where **(a)** follows from M's conditional independence properties (applied to the $\Pr(R_t=r|...)$ and $\Pr(S_{t+1}=x'.s|...)$ terms). Rearranging and taking advantage of marginalization over a (the $\Pr(R_t=r|S_{t+1}=x'.s,...)$ and $\Pr(S_{t+1}=x'.s|...)$ terms can be viewed as a union), we get:

$$
\begin{aligned}
R^i(x,u,x',r) =&\frac{\Pr(S_{t+1}=x'.s|S_t=x.s, U_t^{\text{pre}}=x.u_{\text{pre}}, U_t=u)}{\Pr(S_{t+1}=x'.s|S_t=x.s, U_t^{\text{pre}}=x.u_{\text{pre}}, U_t=u)}\\
&\times\Pr(R_t=r|S_t=x.s, S_{t+1}=x'.s, U_t^{\text{pre}}=x.u_{\text{pre}}, U_t=u)\\
=&\Pr(R_t=r|S_t=x.s, S_{t+1}=x'.s, U_t^{\text{pre}}=x.u_{\text{pre}}, U_t=u)\\
\overset{\text{(a)}}{=}&\Pr(R_t=r|S_t=x.s, U_t^{\text{pre}}=x.u_{\text{pre}}, U_t=u, S_{t+1}=x'.s, U_{t+1}^{\text{pre}}=x'.u_{\text{pre}}),
\end{aligned}
$$

where **(a)** follows from M's conditional independence properties. □

Property 5.

$$\forall s \in \mathcal{S}\ \forall u_{\text{pre}} \in \mathcal{U}_i^{\text{pre}},\ \Pr(\tilde{X}_t = (s, u_{\text{pre}})) = \Pr(S_t = s, U_t^{\text{pre}} = u_{\text{pre}}).$$

Proof.
We present a proof by induction:
Base Case:

$$\begin{aligned}\Pr(S_0 = s, U_0^{\text{pre}} = u_{\text{pre}}) &= \Pr(S_0 = s)\Pr(U_0^{\text{pre}} = u_{\text{pre}} | S_0 = s)\\ &= d_0(s)\pi_i^{\text{pre}}(s, u_{\text{pre}})\\ &= d_0^i((s, u_{\text{pre}}))\\ &= \Pr(\tilde{X}_0^i = (s, u_{\text{pre}})).\end{aligned}$$

Inductive Step:

$$\begin{aligned}\Pr(S_{t+1} = s', U_{t+1}^{\text{pre}} = u'_{\text{pre}}) &\overset{\text{(a)}}{=} \sum_{(s,u_{\text{pre}})\in\mathcal{X}} \Pr(S_t = s, U_t^{\text{pre}} = u_{\text{pre}})\Pr(S_{t+1} = s', U_{t+1}^{\text{pre}} = u'_{\text{pre}} | S_t = s, U_t^{\text{pre}} = u_{\text{pre}})\\ &\overset{\text{(b)}}{=} \sum_{(s,u_{\text{pre}})\in\mathcal{X}} \Pr(S_t = s, U_t^{\text{pre}} = u_{\text{pre}}) \sum_{u\in\mathcal{U}} \Pr(U_t = u | S_t = s, U_t^{\text{pre}} = u_{\text{pre}})\\ &\quad \times \Pr(S_{t+1} = s', U_{t+1}^{\text{pre}} = u'_{\text{pre}} | S_t = s, U_t^{\text{pre}} = u_{\text{pre}}, U_t = u)\\ &\overset{\text{(c)}}{=} \sum_{(s,u_{\text{pre}})\in\mathcal{X}} \Pr(\tilde{X}_t = (s, u_{\text{pre}})) \sum_{u\in\mathcal{U}} \Pr(\tilde{U}_t = u | \tilde{X}_t = (s, u_{\text{pre}}))\\ &\quad \times \Pr(S_{t+1} = s', U_{t+1}^{\text{pre}} = u'_{\text{pre}} | S_t = s, U_t^{\text{pre}} = u_{\text{pre}}, U_t = u),\end{aligned}$$

where **(a)** follows from marginalization over (s, u_{pre}), **(b)** follows from marginalization over u, and **(c)** is through application of the base case and Assumption 1. Notice that the last term is equivalent to P^i by Property 3, which is equivalent to the final term in the next step:

$$\begin{aligned}\Pr(S_{t+1} = s', U_{t+1}^{\text{pre}} = u'_{\text{pre}}) &= \sum_{(s,u_{\text{pre}})\in\mathcal{X}} \Pr(\tilde{X}_t = (s, u_{\text{pre}})) \sum_{u\in\mathcal{U}} \Pr(\tilde{U}_t = u | \tilde{X}_t = (s, u_{\text{pre}}))\\ &\quad \times \Pr(\tilde{X}_{t+1} = (s', u'_{\text{pre}}) | \tilde{X}_t = (s, u_{\text{pre}}), \tilde{U}_t = u)\\ &\overset{\text{(a)}}{=} \sum_{(s,u_{\text{pre}})\in\mathcal{X}} \Pr(\tilde{X}_t = (s, u_{\text{pre}}))\Pr(\tilde{X}_{t+1} = (s', u'_{\text{pre}}) | \tilde{X}_t = (s, u_{\text{pre}}))\\ &\overset{\text{(b)}}{=} \Pr(\tilde{X}_{t+1} = (s', u'_{\text{pre}})),\end{aligned}$$

where **(a)** and **(b)** follow from marginalization over u and (s, u_{pre}), respectively. □

Property 6.

$$\forall s \in \mathcal{S}, \Pr(S_t = s) = \Pr(\tilde{X}_t.s = s).$$

Proof.

$$\begin{aligned}\Pr(S_t = s) &= \sum_{u_{\text{pre}}\in\mathcal{U}_i^{\text{pre}}} \Pr(S_t = s, U_t^{\text{pre}} = u_{\text{pre}})\\ &\overset{\text{(a)}}{=} \sum_{u_{\text{pre}}\in\mathcal{U}_i^{\text{pre}}} \Pr(\tilde{X}_t = (s, u_{\text{pre}}))\\ &\overset{\text{(b)}}{=} \Pr(\tilde{X}_t.s = s),\end{aligned}$$

where **(a)** follows from Property 5 and **(b)** follows from marginalization over u_{pre}. □

Property 7. $\forall s \in \mathcal{S}\ \forall u_{\text{pre}} \in \mathcal{U}_i^{\text{pre}}$, $\pi_i^{\text{pre}}(s, u_{\text{pre}}) = \Pr(\tilde{X}_t.u_{\text{pre}} = u_{\text{pre}} | \tilde{X}_t.s = s)$.

Recall that $\pi_i^{\text{pre}}(s, u_{\text{pre}}) \coloneqq \Pr(U_t^{\text{pre}} = u_{\text{pre}} | S_t = s)$.

Proof.

$$\begin{aligned}
\pi_i^{\text{pre}}(s, u_{\text{pre}}) &= \Pr(U_t^{\text{pre}} = u_{\text{pre}} | S_t = s) \\
&= \frac{\Pr(U_t^{\text{pre}} = u_{\text{pre}}, S_t = s)}{\Pr(S_t = s)} \\
&\overset{\text{(a)}}{=} \frac{\Pr(\tilde{X}_t.u_{\text{pre}} = u_{\text{pre}}, \tilde{X}_t.s = s)}{\Pr(\tilde{X}_t.s = s)} \\
&= \Pr(\tilde{X}_t.u_{\text{pre}} = u_{\text{pre}} | \tilde{X}_t.s = s),
\end{aligned}$$

where **(a)** follows from properties 5 and 6. □

Property 8.

$$\begin{aligned}
&\forall s \in \mathcal{S}\ \forall s' \in \mathcal{S}\ \forall u_{\text{pre}} \in \mathcal{U}_i^{\text{pre}}\ \forall u \in \mathcal{U}, \\
&\Pr(\tilde{X}_{t+1}.s = s' | \tilde{X}_t.s = s, \tilde{X}_t.u_{\text{pre}} = u_{\text{pre}}, \tilde{U}_t = u) = \Pr(S_{t+1} = s' | S_t = s, U_t^{\text{pre}} = u_{\text{pre}}, U_t = u).
\end{aligned}$$

Proof.

$$\begin{aligned}
&\Pr(S_{t+1} = s' | S_t = s, U_t^{\text{pre}} = u_{\text{pre}}, U_t = u) \\
&\overset{\text{(a)}}{=} \sum_{a \in \mathcal{A}} \pi_i^{\text{post}}((s, u_{\text{pre}}), u, a) \Pr(S_{t+1} = s' | S_t = s, U_t^{\text{pre}} = u_{\text{pre}}, U_t = u, A_t = a) \\
&\overset{\text{(b)}}{=} \sum_{a \in \mathcal{A}} \pi_i^{\text{post}}((s, u_{\text{pre}}), u, a) \sum_{u'_{\text{pre}} \in \mathcal{U}^{\text{pre}}} \Pr(U_{t+1}^{\text{pre}} = u'_{\text{pre}} | S_t = s, U_t^{\text{pre}} = u_{\text{pre}}, U_t = u, A_t = a) \\
&\quad \times \Pr(S_{t+1} = s' | S_t = s, U_t^{\text{pre}} = u_{\text{pre}}, U_t = u, A_t = a, U_{t+1}^{\text{pre}} = u'_{\text{pre}}) \\
&\overset{\text{(c)}}{=} \sum_{a \in \mathcal{A}} \pi_i^{\text{post}}((s, u_{\text{pre}}), u, a) \sum_{u'_{\text{pre}} \in \mathcal{U}^{\text{pre}}} \Pr(U_{t+1}^{\text{pre}} = u'_{\text{pre}} | S_t = s, U_t^{\text{pre}} = u_{\text{pre}}, U_t = u, A_t = a, S_{t+1} = s') \\
&\quad \times \Pr(S_{t+1} = s' | S_t = s, U_t^{\text{pre}} = u_{\text{pre}}, U_t = u, A_t = a) \\
&\overset{\text{(d)}}{=} \sum_{a \in \mathcal{A}} \pi_i^{\text{post}}((s, u_{\text{pre}}), u, a) \sum_{u'_{\text{pre}} \in \mathcal{U}^{\text{pre}}} \Pr(U_{t+1}^{\text{pre}} = u'_{\text{pre}} | S_{t+1} = s') \Pr(S_{t+1} = s' | S_t = s, A_t = a),
\end{aligned}$$

where **(a)** follows from marginalization over a and the definition of π_i^{post}, **(b)** follows from marginalization over u'_{pre}, **(c)** follows from the fact that (abbreviating and leaving out the common givens) $\Pr(u'_{\text{pre}}) \Pr(s' | u'_{\text{pre}}) = \Pr(u'_{\text{pre}} | s') \Pr(s')$, and **(d)** follows from M's conditional independence properties (applied to the second and third terms). Notice that the second and third terms above are equivalent to P and π_i^{pre}; plugging those in and rearranging:

$$\begin{aligned}
\Pr(\tilde{X}_{t+1}.s = s' | \tilde{X}_t.s = s, \tilde{X}_t.u_{pre} &= \sum_{u'_{\text{pre}} \in \mathcal{U}^{\text{pre}}} \pi_i^{\text{pre}}(s', u'_{\text{pre}}) \sum_{a \in \mathcal{A}} P(s, a, s') \pi_i^{\text{post}}((s, u_{\text{pre}}), u, a) \\
&\overset{\text{(a)}}{=} \sum_{u'_{\text{pre}} \in \mathcal{U}^{\text{pre}}} P^i((s, u_{\text{pre}}), u, (s', u'_{\text{pre}})) \\
&\overset{\text{(b)}}{=} \sum_{u'_{\text{pre}} \in \mathcal{U}^{\text{pre}}} \Pr(\tilde{X}_{t+1}.u_{\text{pre}} = u'_{\text{pre}} | \tilde{X}_t = (s, u_{\text{pre}}), \tilde{U}_t = u) \\
&\quad \times \Pr(\tilde{X}_{t+1}.s = s' | \tilde{X}_{t+1}.u_{\text{pre}} = u'_{\text{pre}}, \tilde{X}_t = (s, u_{\text{pre}}), \tilde{U}_t = u) \\
&\overset{\text{(c)}}{=} \Pr(\tilde{X}_{t+1}.s = s' | \tilde{X}_t.s = s, \tilde{X}_t.u_{\text{pre}} = u_{\text{pre}}, \tilde{U}_t = u),
\end{aligned}$$

where **(a)** follows from the definition of P^i for the CoMDP, **(b)** follows from the definition of conditional probability, and **(c)** follows from marginalization over u'_{pre}. □

Property 9.

$$\forall s \in \mathcal{S}\ \forall s' \in \mathcal{S}\ \forall u_{\text{pre}} \in \mathcal{U}_i^{\text{pre}}\ \forall u'_{\text{pre}} \in \mathcal{U}_i^{\text{pre}}\ \forall u \in \mathcal{U},$$
$$\Pr(\tilde{X}_{t+1}.u_{\text{pre}} = u'_{\text{pre}} | \tilde{X}_{t+1}.s = s') = \Pr(\tilde{X}_{t+1}.u_{\text{pre}} = u'_{\text{pre}} | \tilde{X}_{t+1}.s = s', \tilde{X}_t = (s, u_{\text{pre}}), \tilde{U}_t = u).$$

Proof.

$$\begin{aligned}
&\Pr(\tilde{X}_{t+1}.u_{\text{pre}} = u'_{\text{pre}} | \tilde{X}_{t+1}.s = s', \tilde{X}_t = (s, u_{\text{pre}}), \tilde{U}_t = u) \\
&= \frac{\Pr(\tilde{X}_{t+1}.u_{\text{pre}} = u'_{\text{pre}}, \tilde{X}_{t+1}.s = s' | \tilde{X}_t = (s, u_{\text{pre}}), \tilde{U}_t = u)}{\Pr(\tilde{X}_{t+1}.s = s' | \tilde{X}_t = (s, u_{\text{pre}}), \tilde{U}_t = u)} \\
&\overset{\text{(a)}}{=} \frac{P^i((s, u_{\text{pre}}), u, (s', u'_{\text{pre}}))}{\Pr(S_{t+1} = s' | S_t = s, U^{\text{pre}} = u_{\text{pre}}, U_t = u)} \\
&= \frac{\pi_i^{\text{pre}}(s', u'_{\text{pre}}) \sum_{a \in \mathcal{A}} P(s, a, s') \pi_i^{\text{post}}((s, u_{\text{pre}}), u, a)}{\Pr(S_{t+1} = s' | S_t = s, U^{\text{pre}} = u_{\text{pre}}, U_t = u)},
\end{aligned}$$

where **(a)** follows from Property 8 applied to the denominator. Expanding the P term and applying M's conditional independence properties:

$$\begin{aligned}
&\Pr(\tilde{X}_{t+1}.u_{\text{pre}} = u'_{\text{pre}} | \tilde{X}_{t+1}.s = s', \tilde{X}_t = (s, u_{\text{pre}}), \tilde{U}_t = u) \\
&= \frac{\pi_i^{\text{pre}}(s', u'_{\text{pre}}) \sum_{a \in \mathcal{A}} \Pr(S_{t+1} = s' | S_t = s, U_t^{\text{pre}} = u_{\text{pre}}, U_t = u, A_t = a) \pi_i^{\text{post}}((s, u_{\text{pre}}), u, a)}{\Pr(S_{t+1} = s' | S_t = s, U^{\text{pre}} = u_{\text{pre}}, U_t = u)} \\
&\overset{\text{(a)}}{=} \frac{\pi_i^{\text{pre}}(s', u'_{\text{pre}}) \Pr(S_{t+1} = s' | S_t = s, U_t^{\text{pre}} = u_{\text{pre}}, U_t = u)}{\Pr(S_{t+1} = s' | S_t = s, U^{\text{pre}} = u_{\text{pre}}, U_t = u)} \\
&= \Pr(\tilde{X}_{t+1}.u_{\text{pre}} = u'_{\text{pre}} | \tilde{X}_{t+1}.s = s'),
\end{aligned}$$

where **(a)** follows from marginalization over a. □

Property 10.

$$\forall r \in \mathcal{R},\ \Pr(R_t = r) = \Pr(\tilde{R}_t^i = r).$$

Proof.

$$\begin{aligned}
\Pr(R_t = r) = &\sum_{s \in \mathcal{S}} \Pr(S_t = s) \sum_{u_{\text{pre}} \in \mathcal{U}^{\text{pre}}} \Pr(U_t^{\text{pre}} = u_{\text{pre}} | S_t = s) \sum_{u \in \mathcal{U}} \Pr(U_t = u | S_t = s, U_t^{\text{pre}} = u_{\text{pre}}) \\
&\times \sum_{s' \in S} \Pr(S_{t+1} = s' | S_t = s, U_t^{\text{pre}} = u_{\text{pre}}, U_t = u) \\
&\times \sum_{u'_{\text{pre}} \in \mathcal{U}^{\text{pre}}} \Pr(U_{t+1}^{\text{pre}} = u'_{\text{pre}} | S_t = s, U_t^{\text{pre}} = u_{\text{pre}}, U_t = u, S_{t+1} = s') \\
&\times \Pr(R_t = r | S_t = s, U_t^{\text{pre}} = u_{\text{pre}}, U_t = u, S_{t+1} = s', U_{t+1}^{\text{pre}} = u'_{\text{pre}}),
\end{aligned}$$

by repeated marginalization. Applying M's conditional independence properties to the $\Pr(U_{t+1}^{\text{pre}}...)$ term:

$$\begin{aligned}
\Pr(R_t = r) = &\sum_{s \in \mathcal{S}} \Pr(S_t = s) \sum_{u_{\text{pre}} \in \mathcal{U}^{\text{pre}}} \Pr(U_t^{\text{pre}} = u_{\text{pre}} | S_t = s) \sum_{u \in \mathcal{U}} \Pr(U_t = u | S_t = s, U_t^{\text{pre}} = u_{\text{pre}}) \\
&\times \sum_{s' \in S} \Pr(S_{t+1} = s' | S_t = s, U_t^{\text{pre}} = u_{\text{pre}}, U_t = u) \sum_{u'_{\text{pre}} \in \mathcal{U}^{\text{pre}}} \Pr(U_{t+1}^{\text{pre}} = u'_{\text{pre}} | S_{t+1} = s') \\
&\times \Pr(R_t = r | S_t = s, U_t^{\text{pre}} = u_{\text{pre}}, U_t = u, S_{t+1} = s', U_{t+1}^{\text{pre}} = u'_{\text{pre}}) \\
\overset{\text{(a)}}{=} &\sum_{s \in \mathcal{S}} \Pr(\tilde{X}_t.s = s) \sum_{u_{\text{pre}} \in \mathcal{U}^{\text{pre}}} \Pr(\tilde{X}_t.u_{\text{pre}} = u_{\text{pre}} | \tilde{X}_t.s = s) \sum_{u \in \mathcal{U}} \Pr(\tilde{U}_t = u | \tilde{X}_t = (s, u_{\text{pre}})) \\
&\times \sum_{s' \in S} \Pr(\tilde{X}_{t+1}.s = s' | \tilde{X}_t = (s, u_{\text{pre}}), \tilde{U}_t = u) \sum_{u'_{\text{pre}} \in \mathcal{U}^{\text{pre}}} \Pr(\tilde{X}_{t+1}.u_{\text{pre}} = u'_{\text{pre}} | \tilde{X}_{t+1}.s = s') \\
&\times \Pr(\tilde{R}_t^i = r | \tilde{X}_t = (s, u_{\text{pre}}), \tilde{U}_t = u, \tilde{X}_{t+1} = (s', u'_{\text{pre}})),
\end{aligned}$$

where **(a)** follows from properties that show various equivalences between the two MDP's. Specifically: Property 6 (first term), Property 7 (second and fifth terms), Assumption 1 (third term), Property 8 (fourth term), and Property 4 (last term). Next, we apply Property 9 to the fifth term:

$$\begin{aligned}
\Pr(R_t = r) = &\sum_{s\in\mathcal{S}} \Pr(\tilde{X}_t.s = s) \sum_{u_{\text{pre}}\in\mathcal{U}^{\text{pre}}} \Pr(\tilde{X}_t.u_{\text{pre}} = u_{\text{pre}}|\tilde{X}_t.s = s) \sum_{u\in\mathcal{U}} \Pr(\tilde{U}_t = u|\tilde{X}_t = (s, u_{\text{pre}})) \\
&\times \sum_{s'\in S} \Pr(\tilde{X}_{t+1}.s = s'|\tilde{X}_t = (s, u_{\text{pre}}), \tilde{U}_t = u) \\
&\times \sum_{u'_{\text{pre}}\in\mathcal{U}^{\text{pre}}} \Pr(\tilde{X}_{t+1}.u_{\text{pre}} = u'_{\text{pre}}|\tilde{X}_{t+1}.s = s', \tilde{X}_t = (s, u_{\text{pre}}), \tilde{U}_t = u) \\
&\times \Pr(\tilde{R}^i_t = r|\tilde{X}_t = (s, u_{\text{pre}}), \tilde{U}_t = u, \tilde{X}_{t+1} = (s', u'_{\text{pre}})) \\
\overset{(a)}{=} &(1)(1)(1)(1)(1) \Pr(\tilde{R}^i_t = r) \\
= &\Pr(\tilde{R}^i_t = r),
\end{aligned}$$

where **(a)** follows from repeated marginalization. □

Lemma 1. *M^i is a Markov decision process.*

Proof. Having defined $\mathcal{X}^i$ as the state set, $\mathcal{U}^i$ as the action set, $\mathcal{R}^i$ as the reward set, P^i as the transition function, R^i as the reward function, d^i_0 as the initial state distribution, and γ_i as the discount parameter, all that remains is to ensure that P^i, R^i, and d^i_0 satisfy their necessary requirements. That is, we must show that these functions are always non-negative and that $\forall x \in \mathcal{X}, \forall u \in \mathcal{U}, \sum_{x'\in\mathcal{X}} P^i(x, u, x') = 1$, $\forall x \in \mathcal{X}, \forall u \in \mathcal{U}, \forall x' \in \mathcal{X}, \sum_{r\in\mathcal{R}^i} \mathcal{R}^i(x, u, x', r) = 1$, and $\sum_{x\in\mathcal{X}} d^i_0(x) = 1$.

The functions are always non-negative because each term in each definition is always non-negative. Next, we show that the sum over the transition function is 1:

$$\begin{aligned}
&\forall x \in \mathcal{X}, \forall u \in \mathcal{U}, \\
\sum_{x'\in\mathcal{X}} P^i(x, u, x') &\overset{(a)}{=} \sum_{x'\in\mathcal{X}} \Pr(S_{t+1} = x'.s, U^{\text{pre}}_{t+1} = x'.u_{\text{pre}}|S_t = x.s, U^{\text{pre}}_t = x.u_{\text{pre}}, U_t = u) \\
&= \sum_{x'.s\in\mathcal{S}} \sum_{x'.u_{\text{pre}}\in U^{\text{pre}}} \Pr(S_{t+1} = x'.s, U^{\text{pre}}_{t+1} = x'.u_{\text{pre}}|S_t = x.s, U^{\text{pre}}_t = x.u_{\text{pre}}, U_t = u) \\
&= 1,
\end{aligned}$$

where **(a)** follows from Property 3. Next, we show that the sum over the reward function is 1:

$$\begin{aligned}
&\forall x \in \mathcal{X},\ \forall u \in \mathcal{U},\ \forall x' \in \mathcal{X}, \\
\sum_{r\in\mathcal{R}^i} R^i(x, u, x', r) &\overset{(a)}{=} \sum_{r\in\mathcal{R}} \Pr(R_t = r|S_t = x.s, U^{\text{pre}}_t = x.u_{\text{pre}}, U_t = u, S_{t+1} = x'.s, U^{\text{pre}}_{t+1} = x'.u_{\text{pre}}) \\
&= 1,
\end{aligned}$$

where **(a)** follows from the fact that $\mathcal{R}^i \coloneqq \mathcal{R}$ and from Property 4.

Finally, we show that the sum of the initial state distribution is 1:

$$\begin{aligned}
\sum_{x\in\mathcal{X}} d^i_0(x) &\overset{(a)}{=} \sum_{x\in\mathcal{X}} d_0(x.s) \underbrace{\pi^{\text{pre}}_i(x.s, x.u_{\text{pre}})}_{=\Pr(U^{\text{pre}}_0 = x.u_{\text{pre}}|S_0 = x.s)} \\
&= \sum_{x.s\in\mathcal{S}} \sum_{x.u_{\text{pre}}\in U^{\text{pre}}} \Pr(S_0 = x.s, U^{\text{pre}}_0 = x.u_{\text{pre}}) \\
&= 1,
\end{aligned}$$

where **(a)** follows from the definition of d^i_0 for the CoMDP.

Therefore, M^i is a Markov decision process. □

Lemma 2. *For all $M, i, \pi^{\text{pre}}_i, \pi^{\text{post}}_i$, and $\bar{\theta}_i$, and given a policy parameterized by θ_i, the corresponding CoMDP M^i satisfies:*

- $\forall x \in \mathcal{X}\ \forall x' \in \mathcal{X}\ \forall u \in \mathcal{U}\ \forall r \in \mathcal{R},\ P^i(x,u,x')$
  $= \Pr(S_{t+1} = x'.s, U^{\text{pre}}_{t+1} = x'.u_{\text{pre}} | S_t = x.s, U^{\text{pre}}_t = x.u_{\text{pre}}, U_t = u)$.
- $\forall x \in \mathcal{X}\ \forall x' \in \mathcal{X}\ \forall u \in \mathcal{U}\ \forall r \in \mathcal{R}$,
  $R^i(x,u,x',r) = \Pr(R_t{=}r | S_t{=}x.s, U^{\text{pre}}_t{=}x.u_{\text{pre}}, U_t{=}u, S_{t+1}{=}x'.s, U^{\text{pre}}_{t+1}{=}x'.u_{\text{pre}})$.
- $\forall s \in \mathcal{S}\ \forall u_{\text{pre}} \in \mathcal{U}^{\text{pre}},\ \Pr(S_t = s, U^{\text{pre}}_t = u_{pre}) = \Pr(\tilde{X}_t = (s, u_{pre}))$.
- $\forall s \in \mathcal{S},\ \Pr(S_t = s) = \Pr(\tilde{X}_t.s = s)$.
- $\forall r \in \mathcal{R},\ \Pr(R_t = r) = \Pr(\tilde{R}^i_t = r)$.

Proof. This follows immediately from properties 3, 4, 5, 6, and 10. □

Property 11. *For all coagents i, for all θ_i, given the same $\theta = (\theta_i, \bar{\theta}_i)$, $J(\theta) = J_i(\theta_i)$.*

Proof.

$$\begin{aligned} J(\theta) &= \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t | \theta\right] \\ &= \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma_i^t R_t^i | \theta_i, \bar{\theta}_i,\right] \\ &= J_i(\theta_i), \end{aligned}$$

where the second step follows directly from Property 10 and the definition of γ_i. □

Lemma 3. *For all coagents i, for all θ_i, $\frac{\partial J_i(\theta_i)}{\partial \theta_i} = \Delta_i(\theta_i)$.*

Proof. In Lemma 1, we proved that the i^{th} CoMDP is an MDP. In Lemma 2, we proved that the i^{th} CoMDP correctly models the i^{th} coagent's environment. Lemma 3 follows directly from these results and the fact that Δ_i is the policy gradient for M^i (Sutton, 2000). □

Theorem 1.
$\nabla J(\theta) = [\Delta_1(\theta_1)^\intercal, \Delta_2(\theta_2)^\intercal, \ldots, \Delta_m(\theta_m)^\intercal]^\intercal$, where m is the number of coagents, and Δ_i is the local policy gradient of the ith coagent.

Proof.

$$\begin{aligned} \nabla J(\theta) &= \left[\frac{\partial J(\theta)}{\partial \theta_1}^\intercal, \frac{\partial J(\theta)}{\partial \theta_2}^\intercal, \ldots, \frac{\partial J(\theta)}{\partial \theta_m}^\intercal\right]^\intercal \\ &\overset{\text{(a)}}{=} \left[\frac{\partial J_1(\theta_1)}{\partial \theta_1}^\intercal, \frac{\partial J_2(\theta_2)}{\partial \theta_2}^\intercal, \ldots, \frac{\partial J_m(\theta_m)}{\partial \theta_m}^\intercal\right]^\intercal \\ &\overset{\text{(b)}}{=} \left[\frac{\Delta_1(\theta_1)}{\partial \theta_1}^\intercal, \frac{\Delta_2(\theta_2)}{\partial \theta_2}^\intercal, \ldots, \frac{\Delta_m(\theta_m)}{\partial \theta_m}^\intercal\right]^\intercal, \end{aligned}$$

where **(a)** follows directly from Property 11 and where **(b)** follows directly from Lemma 3. □

Corollary 2. *If α_t is a deterministic positive stepsize, $\sum_{t=0}^{\infty} \alpha_t = \infty$, $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$, additional technical assumptions are met (Bertsekas & Tsitsiklis, 2000, Proposition 3), and each coagent updates its parameters, θ_i, with an unbiased local policy gradient update $\theta_i \leftarrow \theta_i + \alpha_t \widehat{\Delta}_i(\theta_i)$, then $J(\theta)$ converges to a finite value and $\lim_{t\to\infty} \nabla J(\theta) = 0$.*

Proof. Corollary 2 follows directly from the CPGT, Proposition 3 from Bertsekas & Tsitsiklis (2000), and the assumption that the discounted sum of rewards over an episode is finite (this last assumption prevents $J(\theta)$ from diverging to ∞). □

B Asynchronous Coagent Networks: Supplementary Proofs

B.1 Synchronous Network Correctness

Our goal is to show that the synchronous, acyclic reduction of our original asynchronous, cyclic network behaves identically to our original network. That is, for all $s \in \mathcal{S}, u \in \mathcal{U}^{\text{all}}, c \in \mathbb{N}^m, a \in \mathcal{A}, e \in \{0,1\}^m$, $\grave{\pi}((s,u,c),(a,u',e)) = \Pr(A_t = a, U_t^{\text{all}} = u', E_t = e | S_t = s, U_{t-1}^{\text{all}} = u, C_t = c)$. Because of the large number of variables, if we use one of these lowercase symbols in an equation, assume that it holds for all values in its respective set.

Proof. We present a proof by induction. We assume a topological ordering of the coagents, such that for any $j < i$, the j^{th} coagent executes before the i^{th} coagent. We perform induction over i, with the inductive assumption that the outputs of all the previous coagents, as well as their decisions whether or not to execute, correspond to the original network. The inductive hypothesis is that for all $j < i$:

$$\Pr(\grave{A}_t.u_j^{\text{all}} = u_j', \grave{A}_t.e_j = e_j | \grave{S}_t = (s,u,c)) = \Pr(U_t^{\text{all}}.j = u_j, E_t^j = e_j | S_t = s, U_{t-1}^{\text{all}} = u, C_t = c). \tag{1}$$

Consider the base case, $i = 1$. $\grave{\mathcal{U}}_1^{\text{pre}}$ and $\mathcal{U}_1^{\text{pre}}$ are both the empty set, because no coagents produce an output before the first coagent in either network. As a result, the distribution over the execution probability is trivially the same in both networks, that is, $\Pr(E_t^1 = 1 | S_t = s, U_{t-1}^{\text{all}} = u, C_t = c) = \beta_1((s,\varnothing,u), c_1) = \Pr(\grave{A}_t.e_1 = 1 | \grave{S}_t = (s,u,c))$. Next, we consider the action. If the coagent executes, $\Pr(U_t^i = u_i' | E_t^1 = 1, S_t = s, U_{t-1}^{\text{all}} = u, C_t = c) = \pi_1((s,\varnothing,u), u_i') = \Pr(\grave{A}_t.u_i^{\text{all}} = u_i' | A_t.e_i = 1, \grave{S}_t = (s,u,c))$. If the coagent does not execute, the action is trivially u_i in both cases. Therefore, Equation (1) holds for $j = 1$.

Next we consider the inductive step, where we show that Equation (1) holds for the i^{th} coagent given that it holds for $j < i$. First we consider the execution function, the output of which is represented in the synchronous setting by $\grave{A}_t.e_i$, and in the asynchronous setting by E_t. In the asynchronous setting, the probability of the i^{th} coagent executing at time step t is $\beta_i((S_t, U_t^{\text{pre}}, U_{t-1}^{\text{all}}), C_t^i)$. Since we are not given U_t^{pre}, we must sum over possible values:

$$\Pr(E_t^i = 1 | S_t = s, U_t^{\text{all}} = u, C_t = c) = \sum_{u_{\text{pre}} \in \mathcal{U}_i^{\text{pre}}} \beta_i((s, u_{\text{pre}}, u), c_i) \Pr(U_t^{\text{pre}} = u_{\text{pre}} | S_t = s, U_{t-1}^{\text{all}} = u, C_t = c)$$

In the reduced setting, we instead have a coagent, such that $\Pr(\grave{A}_t.e_i = 1 | \grave{S}_t = (s,u,c)) = \beta_i((s, \grave{U}_t^{\text{pre}}, u), c_i)$. Again, we sum over possible values of $\grave{U}_t^{\text{pre}}$:

$$\Pr(\grave{A}_t.e_i = 1 | \grave{S}_t = (s,u,c)) = \sum_{u_{\text{pre}} \in \mathcal{U}_i^{\text{pre}}} \beta_i((s, u_{\text{pre}}, u), c_i) \Pr(\grave{U}_t^{\text{pre}}.u = u_{\text{pre}} | \grave{S}_t = (s,u,c)).$$

Recall the reduced setting was defined such that for all $j < i$, $\grave{A}_t.u_j^{\text{all}} = \grave{U}_t^{\text{pre}}.u_j$, and in the asynchronous setting, $U_t^{\text{pre}}.u_j = U_t^{\text{all}}.u_j$. We therefore can conclude from (1) and by substitution that for all $j < i, \Pr(\grave{U}_t^{\text{pre}}.u_j = u | \grave{S}_t = (s,u,c)) = \Pr(U_t^{\text{pre}}.u_j = u | S_t = s, U_{t-1}^{\text{all}} = u, C_t = c)$. Substituting this into the above equations:

$$\begin{aligned}\Pr(\grave{A}_t.e_i = 1 | \grave{S}_t = (s,u,c)) &= \sum_{u_{\text{pre}} \in \mathcal{U}_i^{\text{pre}}} \beta_i((s, u_{\text{pre}}, u), c_i) \Pr(U_t^{\text{pre}} = u_{\text{pre}} | S_t = s, U_{t-1}^{\text{all}} = u, C_t = c) \\ &= \Pr(E_t^i = 1 | S_t = s, U_t^{\text{all}} = u, C_t = c).\end{aligned}$$

Note also that from the perspective of $\grave{\pi}_i$, $\grave{A}_t.e_i = \grave{U}_t^{\text{pre}}.e_i$. Next we consider the output of the i^{th} coagent, given in the asynchronous setting as U_t^i, and in the reduced setting by $\grave{A}_t.u_i^{\text{all}}$. In the original setting, U_t^i was given such that for all $u_i \in U^i$:

$$\Pr(U_t^i = u_i' | S_t = s, U_{t-1}^{\text{all}} = u, C_t = c, U_t^{\text{pre}} = u_{\text{pre}}, E_t^i = e_i) = \begin{cases} \pi_i((s, u_{\text{pre}}, u), u_i'), & \text{if } e_i = 1 \\ 1, & \text{if } e_i = 0 \text{ and } u_i' = u_i \\ 0, & \text{otherwise.} \end{cases}$$

In the synchronous setting, we are given:

$$\begin{aligned}\Pr(\grave{A}_t.u_i^{\text{all}} = u_i' | \grave{S}_t = (s,u,c), \grave{U}_t^{\text{pre}}.u = u_{\text{pre}}, \grave{U}_t^{\text{pre}}.e_i = e_i) &= \grave{\pi}_i(((s,u,c), u_{\text{pre}}), u) \\ &= \begin{cases} \pi_i((s, u_{\text{pre}}, u), u_i'), & \text{if } e_i = 1 \\ 1, & \text{if } e_i = 0 \text{ and } u_i' = u_i \\ 0, & \text{otherwise.} \end{cases}\end{aligned}$$

Since we were given s, u, and c, assumed through the inductive hypothesis that $\Pr(U_t^{\text{pre}}.u = u_{\text{pre}}|S_t = s, U_{t-1}^{\text{all}} = u, C_t = c) = \Pr(U_t^{\text{pre}} = u_{\text{pre}}|S_t = s, U_{t-1}^{\text{all}} = u, C_t = c)$, and showed that $\Pr(U_t^{\text{pre}}.e_i = e_i|U_t^{\text{pre}}.u = u_{\text{pre}}, S_t = s, U_{t-1}^{\text{all}} = u, C_t = c) = \Pr(E_t^i|U_t^{\text{pre}} = u_{\text{pre}}, S_t = s, U_{t-1}^{\text{all}} = u, C_t = c)$, we know that the distributions over the variables we conditioned on are equal. Since we also showed that the conditional distributions are equal, we conclude that $\Pr(\grave{A}_t.u_i^{\text{all}} = u_i'|\grave{S}_t = (s, u, c)) = \Pr(U_t^i = u_i'|S_t = s, U_{t-1}^{\text{all}} = u, C_t = c)$.

This completes the inductive proof that $\Pr(\grave{A}_t.u^{\text{all}} = u', \grave{A}_t.e = e|\grave{S}_t = (s, u, c)) = \Pr(U_t^{\text{all}} = u', E_t = e|S_t = s, U_{t-1}^{\text{all}} = u, C_t = c)$. We still must consider $\grave{A}_t.a$. This is given by the output of some predefined subset of coagents, which is the same subset in both the synchronous and asynchronous network. We showed that the distribution over outputs was the same for corresponding coagents in the two networks, and therefore can conclude immediately that $Pr(\grave{A}_t.a = a|\grave{S}_t = (s, u, c)) = \Pr(A_t = a|S_t = s, U_{t-1}^{\text{all}} = u, C_t = c)$. Finally:

$$\begin{aligned}\Pr(A_t = a, U_t^{\text{all}} = u', E_t = e|S_t = s, U_{t-1}^{\text{all}} = u, C_t = c) &= \Pr(\grave{A}_t.a = a, \grave{A}_t.u^{\text{all}} = u', \grave{A}_t.e = e|\grave{S}_t = (s, u, c)) \\ &= \Pr(\grave{A}_t = (a, u', e)|\grave{S}_t = (s, u, c)) \\ &= \grave{\pi}((s, u, c), (a, u', e)).\end{aligned}$$

□

B.2 Equivalence of Objectives

In both settings, the network depends on the same parameter vector, θ. In this section, we show that for all settings of this parameter vector the resulting sum of rewards is equivalent in both settings. That is, $J(\theta) = \grave{J}(\theta)$.

Proof. We begin by showing that the distribution over the "true" states and actions is equal in both settings, that is, for all $s \in \mathcal{S}, a \in \mathcal{A}$, $Pr(\grave{S}_t.s = s, \grave{A}_t.a = a) = \Pr(S_t = s, A_t = a)$. Once this is shown, we show that the reward distributions are the same, that is, for all r, $\Pr(\grave{R}_t = r) = \Pr(R_t = r)$. Finally, we show $J(\theta) = \grave{J}(\theta)$.

B.2.1 Equivalence of State Distributions

First, we show that $\Pr(\grave{S}_t = (s, u, c)) = \Pr(S_t = s, U_{t-1}^{\text{all}} = u, C_t = c)$, by induction over time steps. The base case is the initial state, $\grave{S}_0$. We know from the definition of $\grave{d}_0$ that for all i, $\grave{S}_0.c_i = C_0^i = 0$, so we will focus on S_0 and U_{-1}^{all}. We assumed in the problem setup for the asynchronous setting that for all i and j, the random variables S_0, U_{-1}^i, and U_{-1}^j are independent. Where c is the zero vector, for all s and u:

$$\begin{aligned}\Pr(\grave{S}_0 = (s, u, c)) &= \grave{d}_0((s, u, c)) \\ &= d_0(s) \prod_{i=1}^{m} h_0^i(u_i) \\ &= \Pr(S_0 = s) \prod_{i=1}^{m} \Pr(U_{-1}^i = u_i) \\ &= \Pr(S_0 = s, U_{-1}^{\text{all}} = u) \\ &= \Pr(S_0 = s, U_{-1}^{\text{all}} = u, C_0 = c).\end{aligned}$$

Thus, we've proven the base case. Next we consider the inductive step:

$$\begin{aligned}&\Pr(\grave{S}_{t+1} = (s', u', c')|\grave{S}_t = (s, u, c)) \\ &= \sum_{(a,u'',e)\in\grave{\mathcal{A}}} \Pr(\grave{S}_{t+1} = (s', u', c')|\grave{S}_t = (s, u, c), \grave{A}_t = (a, u'', e)) \Pr(\grave{A}_t = (a, u'', e)|\grave{S}_t = (s, u, c)) \\ &= \sum_{(a,u'',e)\in\grave{\mathcal{A}}} \grave{P}((s, u, c), (a, u'', e), (s', u', c'))\grave{\pi}((s, u, c), (a, u'', e)) \\ &= \sum_{a\in\mathcal{A}, e\in\mathcal{E}} \begin{cases} P(s, a, s')\grave{\pi}((s, u, c), (a, u', e)) & \text{if } f_c(c, e) = c', u' = u'' \\ 0 & \text{otherwise,}\end{cases}\end{aligned}$$

The case statement comes from the definition of $\grave{P}$. Given c and c', there is a unique value, which we call e', that satisfies $f_c(c, e) = c'$. If $u' \neq u''$ or $e \neq e'$, the inner expression is zero. Therefore:

$$\Pr(\grave{S}_{t+1} = (s', u', c') | \grave{S}_t = (s, u, c)) = \sum_{a \in \mathcal{A}} P(s, a, s') \grave{\pi}((s, u, c), (a, u', e')).$$

Next, we can apply the equivalence shown in section B.1 and the definition of P:

$$\begin{aligned}
&\sum_{a \in \mathcal{A}} P(s, a, s') \grave{\pi}((s, u, c), (a, u', e')) \\
=&\sum_{a \in \mathcal{A}} \Pr(S_{t+1} = s' | A_t = a, S_t = s) \Pr(A_t = a, U_t^{\text{all}} = u', E_t = e' | S_t = s, U_{t-1}^{\text{all}} = u, C_t = c).
\end{aligned}$$

Remember that $C_{t+1} = f_c(C_t, E_t)$. Therefore, we can exchange $\Pr(E_t = e' | C_t = c)$ for $\Pr(C_{t+1} = c' | C_t = c)$:

$$\begin{aligned}
&\sum_{a \in \mathcal{A}} \Pr(S_{t+1} = s' | A_t = a, S_t = s, E_t = e, C_t = c) \Pr(A_t = a, U_t^{\text{all}} = u', C_{t+1} = c' | S_t = s, U_{t-1}^{\text{all}} = u, C_t = c) \\
=&\sum_{a \in \mathcal{A}} \Pr(S_{t+1} = s', A_t = a, U_t^{\text{all}} = u', E_t = e | S_t = s, U_{t-1}^{\text{all}} = u, C_t = c) \\
=&\Pr(S_{t+1} = s', U_t^{\text{all}} = u', C_{t+1} = c' | S_t = s, U_{t-1}^{\text{all}} = u, C_t = c).
\end{aligned}$$

Inductively, we have shown that for all t, $\Pr(\grave{S}_t = (s, u, c)) = \Pr(S_t = s, U_{t-1}^{\text{all}} = u, C_t = c)$.

B.2.2 Equivalence of Reward Distributions

It follows immediately from the above equality and B.1 that $\Pr(\grave{A}_t = (a, u, e)) = \Pr(A_t = a, U_t^{\text{all}} = u, E_t = e)$. We turn our attention to the reward distribution:

$$\begin{aligned}
&\Pr(\grave{R}_t = r) \\
=&\sum_{(s,u,c) \in \grave{\mathcal{S}}} \sum_{(a,u',e) \in \grave{\mathcal{A}}} \sum_{(s',u'',c') \in \grave{\mathcal{S}}} \Pr(\grave{R}_t = r | \grave{S}_t = (s, u, c), \grave{A}_t = (a, u', e), \grave{S}_{t+1} = (s', u'', c')) \\
&\times \Pr(\grave{S}_t = (s, u, c), \grave{A}_t = (a, u', e), \grave{S}_{t+1} = (s', u'', c')) \\
=&\sum_{(s,u,c) \in \grave{\mathcal{S}}} \sum_{(a,u',e) \in \grave{\mathcal{A}}} \sum_{(s',u'',c') \in \grave{\mathcal{S}}} \grave{R}((s, u, c), (a, u', e), (s', u'', c')) \Pr(\grave{S}_t = (s, u, c), \grave{A}_t = (a, u', e), \grave{S}_{t+1} = (s', u'', c')) \\
=&\sum_{(s,u,c) \in \grave{\mathcal{S}}} \sum_{(a,u',e) \in \grave{\mathcal{A}}} \sum_{(s',u'',c') \in \grave{\mathcal{S}}} R(s, a, s') \Pr(\grave{S}_t = (s, u, c), \grave{A}_t = (a, u', e), \grave{S}_{t+1} = (s', u'', c')) \\
=&\sum_{(s,u,c) \in \grave{\mathcal{S}}} \sum_{(a,u',e) \in \grave{\mathcal{A}}} \sum_{(s',u'',c') \in \grave{\mathcal{S}}} R(s, a, s') \Pr(S_t = s, U_{t-1}^{\text{all}} = u, C_t = c, A_t = a, U_t^{\text{all}} = u', E_t = e, S_{t+1} = s', C_{t+1} = c') \\
=&\sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} R(s, a, s') \Pr(S_t = s, A_t = a, S_{t+1} = s') \\
=&\sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \Pr(R_t = r | S_t = s, A_t = a, S_{t+1} = s') \Pr(S_t = s, A_t = a, S_{t+1} = s') \\
=&\Pr(R_t = r).
\end{aligned}$$

B.2.3 Equivalence of Objectives

Finally, we show the objectives are equal:

$$\grave{J}(\theta) = \mathbf{E}[\sum_{t=0}^{T} \grave{\gamma}^t \grave{R}_t] = \mathbf{E}[\sum_{t=0}^{T} \gamma^t R_t] = J(\theta),$$

by linearity of expectation. □

C Experimental Details of Finite Difference Comparison

We use a simple 3×3 Gridworld. The network structure used in this experiment consists of three coagents with tabular state-action value functions and softmax policies: Two receive the tabular state as input, and each of those two coagents have a single tabular binary output to the third coagent, which in turn outputs the action (up, down, left, or right). This results in two coagents with 18 parameters each, and one coagent with 16 parameters, resulting in a network with 52 parameters. The coagents asynchronously execute using a geometric distribution; the environment updates every step and each coagent has a 0.5 probability of executing each step. The gradient estimates appear to converge, providing empirical support of the CPGT. The data is drawn from 20 trials. 5×10^8 episodes were used for each finite difference estimate. For each trial, five training episodes were conducted before the parameters were frozen and the two gradient estimation methods were run. The coagents were trained with Sutton & Barto's (2018) actor-critic with eligibility traces algorithm and shared a single critic. Note that the critic played no role in the gradient estimation methods, only in the initial training episodes. Hyperparmaters used: critic step size = 0.024686, $\gamma = 1$, input agent step size = 0.02842, output agent step size = 0.1598, and all agents' $\lambda = 0.8085$.