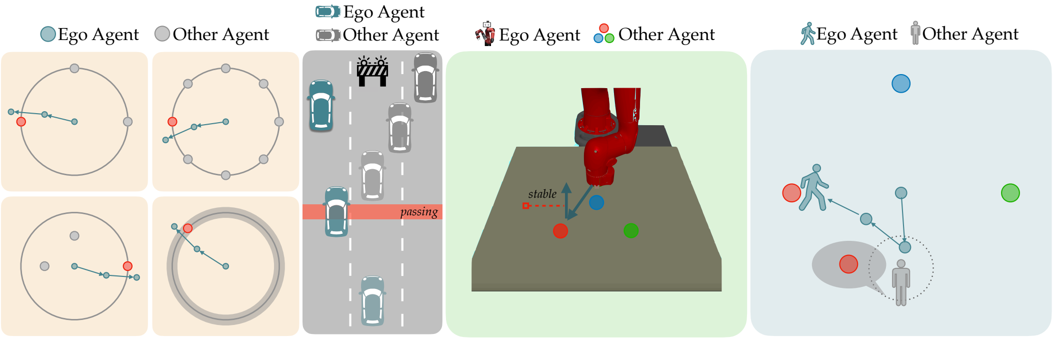

Circle Point Mass Driving Sawyer-Reach Detour Speaker-Listener

Figure 4: Stabilizing behaviors in all simulated environments. In the Circle Point Mass environments, the ego agent ends each interaction outside the circle to stabilize the opponent's strategy. In the Driving environment, the ego agent passes on the left before the red line, establishing the convention that faster vehicles pass on the left. In Sawyer-Reach, the ego agent ends the interaction above a fixed plane in the z-axis to stabilize the opponent's choice of goal location. In Detour Speaker-Listener, the ego agent takes a detour to go near the speaker to stabilize the opponent's communication strategy.

In the Appendix, we provide specific implementation details with relevant hyperparameters, and we also describe further details of our simulated environments, including a discussion about the types of opponent strategies we consider. We provide qualitative analysis of trajectories from interactions at `https://sites.google.com/view/stable-marl/`.

A Implementation Details

Both the encoder and decoder are implemented as multilayer perceptrons (MLPs) with 2 fully connected layers of hidden size 256. The encoder outputs the predicted latent strategy z^{j+1} with latent dimension 10 given tuples of (s, a, r, s') from the previous interaction τ^j. The decoder then reconstructs the next states and rewards given the states, actions, and predicted latent strategy. To train the ego agent's policy, we use soft actor-critic (SAC) [45]. Both the actor π and critic Q are MLPs with 2 fully connected layers of hidden size 256 conditioned on state s and predicted latent strategies z^j, z^{j-1}. With a modified reward function, we use the same parameter update equations as [10] to update the encoder, decoder, actor, and critic. For the combination of task and stability reward, we used an equal weighting of $\beta = 0.5$.

Discrete Latent Variables. For the discrete latent variables, we use temperature $\lambda = 1$ for the Gumbel-Softmax computation. We use the Straight-Through Gumbel-Estimator, where the latent representation z is discretized using $\arg\max$ for the forward pass but the continuous approximation is used for the backward pass. Concretely, the Gumbel-Max sampling method allows us to produce latent strategies from a categorical distribution with probabilities produced by the encoder:

$$z^j = \text{one\_hot}(\arg\max_i [g_i + \log \mathcal{E}_\phi(\tau^{j-1})_i], \tag{3}$$

where $g_1 \dots g_k \sim \text{Gumbel}(0, 1)$ [48, 49]. For the backward pass, the continuous, differentiable approximation with temperature λ is as follows:

$$\tilde{z}^j_i = \frac{e^{[g_i + \log \mathcal{E}_\phi(\tau^{j-1})_i]/\lambda}}{\sum_{a=1}^k e^{[g_a + \log \mathcal{E}_\phi(\tau^{j-1})_a]/\lambda}} \tag{4}$$

B Further Details of Simulated Environments

The simulated environments with trajectories of stabilizing behavior are shown in Fig. 4.

Circle Point Mass. In the Circle Point Mass environments, the ego agent observes its own position and does not observe the opponent's position. The ego agent has a continuous 2-dimensional action space determining the agent's velocity. The task reward is defined as the Euclidean distance between the ego and opponent position: $\mathcal{R}_{\text{task}} = -||s - z||_2$. The oracle observes the opponent's position. The interaction ends after 50 timesteps.

Driving. In the Driving environment, the ego agent tries to consistently pass the opponent. The ego agent observes both their own position and the opponent's position, but does not observe the opponent's strategy or intent of which lane they will merge towards. The ego agent has a continuous 1-dimensional action space determining the agent's lateral velocity. The task reward is defined as -1 if a collision occurs 0 if no collision occurs. The oracle observes the opponent's intended choice of lane. The interaction ends after 10 timesteps.

Sawyer-Reach. In the Sawyer-Reach environment, the ego agent must try to reach one of three opponent's positions on a table. The ego agent observes their own end-effector position and does not observe the opponent's position. The ego agent has a continuous 3-dimensional action space determining the end effectors's velocity through the use of a mocap. The reward function is defined as the Euclidean distance between the ego agent's end-effector and the opponent's position: $\mathcal{R}_{\text{task}} = -||s - z||_2$. The oracle observes the opponent's position and the interaction ends after 50 timesteps. Unlike in the Circle Point Mass, the movement of the ego agent's end-effector is limited by the simulated physical limits of the Sawyer robot.

Detour Speaker-Listener. In the Detour Speaker-Listener environment, the ego agent must try to reach one of three unknown goal locations by learning how to interpret the communication of the opponent speaker. The ego agent observes their own position as well as the communication from the opponent. The opponent observes the true underlying goal and uses their strategy to decide how to communicate the goal to the ego agent. The ego agent has a continuous 2-dimensional action space determining the agent's velocity. The reward function is defined as the Euclidean distance between the ego agent's position and the opponent's position: $\mathcal{R}_{\text{task}} = -||s - z||_2$. The oracle observes the opponent position and the interaction ends after 50 timesteps. If the ego agent has not gone near the speaker within the current interaction, the opponent's communication strategy changes with probability 0.5.

In this environment, ideal behavior would be to stabilize the opponent into maintaining the same communication method, so learning how to map the communication to the correct goal landmark becomes easy. If the ego listener agent cannot interpret the speaker's communication, the listener would likely go to the average position (centroid) of all possible opponent positions.

C Equally Beneficial Latent Strategies.

We mainly consider settings where each latent strategy is equally optimal in terms of the task reward, meaning for any fixed start state $s_1 \in \mathcal{S}$, $V^*(s_1, z)$ is constant for all $z \in \mathcal{Z}$. However, it may not be possible to stabilize to all strategies. While our approach can be applied to scenarios where some latent strategies are better than others, in many multi-agent environments, there are multiple possible equally optimal opponent strategies, and the symmetry of the strategies causes the opponent to indecisively switch between strategies in an unstable manner. Thus, stabilizing in these environments can be seen as a symmetry-breaking method to influence the opponent to stay at a stable strategy, forming multi-agent conventions [50]. We show promising results in one setting with unequally beneficial latent strategies (see row 3, Fig. 3). Future work should further consider environments where there are unequally beneficial strategies for each agent.

D Effect of the Stability Weight

The stability weight β governs the relative importance of optimizing for the stability reward versus the task reward. We investigate the effect of choosing a stability weight on SILI's performance in Fig. 5 with the Circle (3 Goals) environment. We vary β from 0 to 1, where $\beta = 0$ is an instance where the agent only optimizes for the task reward and $\beta = 1$ is where the agent only optimizes for the stability reward. From Fig. 5, we see that the agent learns how to stabilize the opponent's strategy as long as β is large enough ($\beta \geq 0.4$). However, setting β too large can result in the agent optimizing less for the task reward. There is an implicit ordering in which the ego agent should first learn to stabilize the opponent's strategy, and then consequently learn a best response to the

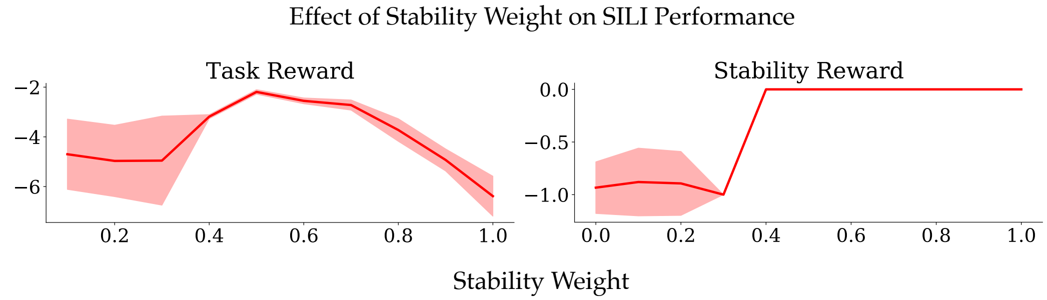

Figure 5: Effect of the stability weight on the task and stability reward in the Circle (3 Goals) environment. (Left) There appears to be a sweet spot between $0.4 \leq \beta \leq 0.8$ where SILI achieves the largest task reward. When β is too small, the agent does not learn to stabilize the opponent's strategy, so the task reward remains sub-optimal. When β is too large, the agent focuses only on stabilizing the opponent's strategy without trying to optimize for the task reward. (Right) When $\beta \geq 0.4$, the stability reward is maximized. This suggests that in practice, annealing the stability weight β may be beneficial, as the agent can first learn to stabilize the opponent's strategy, and then learn to maximize the task reward with respect to a stable opponent strategy.

fixed opponent's strategy. Thus, in practice, we found that annealing the stability weight can be an effective method to avoid needing to tune the β hyperparameter.