A. Game Theory

In the context of this paper, a **strategy** of a primitive ω is equivalent to its bidding policy ψ. A **strategy profile** is the set of strategies $\psi^{1:N}$ for all primitives $\omega^{1:N}$. For emphasis, we equivalently write the utility $U^i\left(\psi^{1:N}\right)$ for player i as $U^i\left(\psi^i;\psi^{-i}\right)$, where ψ^i is the strategy for player i and ψ^{-i} is the strategy for all other players.

Definition A.1. ***Best Response****: A strategy ψ^i is the best response for player i if given the strategies of all other players ψ^{-i}, ψ^i maximizes player i's utility $U^i\left(\psi^i;\psi^{-i}\right)$.*

Definition A.2. ***Nash Equilibrium****: A strategy profile is in a Nash equilibrium if given the strategies of other players, each player's strategy is a best response.*

Definition A.3. ***Dominant Strategy Equilibrium****: A strategy ψ is a dominant strategy if it is the best response for a player no matter what strategies the other players play. A dominant strategy equilibrium is the unique Nash equilibrium where every player plays their dominant strategy.*

Definition A.4. ***Weakly Dominated Strategies****: Consider player i playing strategy ψ^i. Let the strategies for all other players be ψ^{-i}. A strategy ψ^i* ***weakly dominates*** *$\tilde{\psi}^i$ if for all strategies of other players ψ^{-i}, $U^i(\psi;\psi^{-i}) \geq U^i(\tilde{\psi};\psi^{-i})$, and there exists a $\tilde{\psi}^{-i}$ such that $U^i(\psi^i;\tilde{\psi}^{-i}) > U^i(\tilde{\psi}^i;\tilde{\psi}^{-i})$. ψ is* ***dominated*** *if there exists a strategy which dominates it.*

Definition A.5. ***Iterated Deletion of Dominated Strategies****: Iterated deletion of dominated strategies repeatedly (a) removes all dominated strategies for each player, then (b) updates the dominance relation (as now there are fewer strategies of other players to consider). It terminates when all remaining strategies are undominated.*

Vickrey Auction The Vickrey Auction is a type of single-item sealed bid auction. It is single-item, which means there is a single auction item up for sale. It is sealed-bid, which means that the players have no knowledge of each others' bids.

B. Societal Decision-Making Framework

Abstraction Level	Global	Local
Agent	Society $\Omega = \{\omega^i\}_{i=1}^N$	Primitive $\omega^i = \left(\psi^i, \phi^i\right)$
Environment	global MDP	local auction at state s
State Space	$\mathcal{S}$	the single state $\{s\}$
Action Space	the indices of the primitives $\mathcal{A} = \{1, 2, ..., N\}$	the space of non-negative bids $\mathcal{B} = \mathbb{R}_{\geq 0}$
Objective	$J(\pi_\Omega) = \mathbb{E}_{\tau\sim p^\pi(\tau)}\left[\sum_{t=0}^T \gamma^t r\left(s_t, \omega_t\right)\right]$	$U_s^i(\psi^{1:N}) = \mathbf{v}_s^i \cdot X^i(\mathbf{b}) - P^i(\mathbf{b})$
Problem	$\max_{\pi_\Omega} J(\pi_\Omega)$	$\max_{\psi^i} U_s^i(\psi^{1:N})$

Table 1. **Societal Decision-Making.** This table specifies the **agent**, **environment**, **objective**, and **problem** at both the **global** and **local** levels of abstraction in the societal-decision-making framework.

Though both the monolithic decision-making framework as well as the societal decision-making (in which the transformations ϕ correspond to literal actions) can both be used for reinforcement learning, the key difference between the two is that the learnable parameters are trained to optimize the objective of the MDP in the monolithic framework, whereas the learnable parameters are trained to optimize the objective of the auction at each state of the MDP in the societal framework.

C. The Cloned Vickrey Society as a Solution to Societal Decision-Making

The MDP specifies the environment. The society specifies the abstract agent that interacts in the environment. The **Market MDP** is a global MDP governed by a specific type of auction mechanism (the Vickrey mechanism) that is agnostic to the architecture of the society that interacts with it. A **cloned society** is a specific architecture of society (one with redundant

primitives) that is agnostic to the auction mechanism governing the global MDP. The **cloned Vickrey society** specifies a specific architecture of a society (a cloned society) that interacts with a specific type of global MDP (a Market MDP).

C.1. Proofs

Proposition 5.1. *Assume at each state s the local auction allocates $X^i(\mathbf{b}) = 1$ if i wins and $X^i(\mathbf{b}) = 0$ if i loses. Then all primitives ω^i bidding their optimal societal Q-values $Q^*_\Omega(s, \omega^i)$ collectively induce an optimal global policy.*

Proof. For state s, the index of the primitive with the highest bid is equal to the primitive with the highest optimal societal Q-value for that state. That is, $\arg\max_i \mathbf{b}^i_t = \arg\max_i Q^*_\Omega(s_t, \omega^i_t)$. Thus, selecting the highest-bidding primitive to win by definition follows the optimal policy for the Market MDP. □

Theorem 5.2 *If the valuations $\mathbf{v}^i_s$ for each state s are the optimal societal Q-values $Q^*_\Omega(s, \omega^i)$, then the society's optimal global policy coincides with the primitives' unique dominant strategy equilibrium under the Vickrey mechanism.*

Proof. By defining the valuation of each primitive ω^i to be its optimal societal Q-value $Q^*_\Omega(s_t, \omega^i_t)$, the DSIC property of the Vickrey auction guarantees the dominant strategy equilibrium is to bid exactly $Q^*_\Omega(s_t, \omega^i_t)$. The welfare-maximization property of the Vickrey auction guarantees if all primitives played their dominant strategies, then the primitive with the highest valuation wins, so specifying the valuations such that activating the primitive with the highest valuation at time t follows the optimal global policy at that timestep. □

Proposition 5.3. *In a Market MDP, it is a Nash equilibrium for every primitive to bid $Q^*_\Omega(s, \omega^i)$. Moreover, if the Market MDP is finite horizon, then bidding $Q^*_\Omega(s, \omega^i)$ is the unique Nash equilibrium that survives iterated deletion of weakly dominated strategies.*

Proof. Consider time-step t. Assuming that every other primitive at every other time-step bids Q^*_Ω, the best response under the Vickrey mechanism for primitive ω^i at timestep t would be to truthfully bid $r(s_t, \omega^i_t) + \gamma \cdot \max_k Q^*_\Omega(s_{t+1}, \omega^k)$, which by definition of the Bellman optimality equations (Bellmann, 1957) is $Q^*_\Omega(s_t, \omega^i_t)$.

For the finite horizon case, we proceed by backwards induction.

Base Case: The last time-step T is a bandit problem where $Q^*_\Omega(s_T, \omega^i_T) = r(s_T, \omega^i_T)$, so by Theorem 5.2 bidding $Q^*_\Omega(s_t, \omega^i_t)$ is the unique dominant strategy.

Inductive Hypothesis: In time-step $t + 1$, the strategy which survives iterated deletion of dominated strategies is to bid the optimal societal Q-value.

Inductive Step: By the Inductive Hypothesis, in time-step $t + 1$, all primitives bid their optimal societal Q-values if they use any strategy which survives iterated deletion of dominated strategies. This means that in time-step t, each primitive's valuation for winning is given exactly by their optimal societal Q-value: $r(s_t, \omega^i_t) + \gamma \cdot \max_k Q^*_\Omega(s_{t+1}, \omega^k)$. Therefore, it is a dominant strategy to bid Q^*_Ω, and all other bids are dominated (and therefore removed). □

Lemma 5.4 *For a cloned society, at the Nash equilibrium specified in Proposition 5.3, what the winning primitive $\hat{\omega}^i$ at time t receives from the winning primitive $\hat{\omega}^k$ at $t + 1$ is exactly what $\hat{\omega}^k$ pays: $\mathbf{b}^k_{s_{t+1}}$.*

Proof. If two primitives have the same ϕ, then their societal Q-values are identical, so their optimal strategies ψ are identical. Then at the Nash equilibrium specified in Proposition 5.3, their bids are also identical. □

Theorem 5.5. *Define a **cloned Vickrey society** as a cloned society that solves a Market MDP. Then it is a Nash equilibrium for every primitive in the cloned Vickrey society to bid $Q^*_\Omega(s, \omega^i)$. In addition, the price that the winning primitive pays for winning is equivalent to what it bid.*

Proof. (1) is by Proposition 5.3 and (2) is by Lemma 5.4. □

D. Decentralized Reinforcement Learning Algorithms for the Cloned Vickrey Society

Section 5 presented the cloned Vickrey society as a concrete instantiation of the societal decision-making framework whose optimal global policy emerges as a Nash equilibrium of self-interested primitives engaging in local economic transactions. Section 6 removed the assumption that primitives know their valuations and presented decentralized RL algorithms for learning these valuations through interaction.

In this paper, we specified the class of decentralized RL algorithms by the learning objective – the set of local auction utilities at every state. We present in Algorithm 1 the algorithm pseudocode for an on-policy variant of such a decentralized RL algorithm, but the learning objective is in principle agnostic to the RL algorithm use to train each primitive. However, simply specifying only one variant within this class of algorithm in this paper leaves much to still be explored. Adapting methods developed in the monolithic framework, including bandit algorithms, off-policy algorithms, and on-policy algorithms, for optimizing the local auction utilities the would be an interesting direction for future work.

Algorithm 1 On-Policy Decentralized RL

Initialize: Primitives $\omega^{1:N}$, Memory $m^{1:N}$, RL update rule f
while True **do**
  ▷ *Sample Episode*
  while episode has not terminated **do**
    $\omega^{1:N}$ observe state s
    $\omega^{1:N}$ produce bids $\mathbf{b}_s^1, ..., \mathbf{b}_s^N$
    Auction selects winner $\hat{\omega}$ with transformation $\hat{\phi}_{\mathcal{T}}$
    $\hat{\omega}$ produces $s' = \hat{\phi}_{\mathcal{T}}(s)$
    Record environment reward $r(s, \hat{\omega})$
    $s \leftarrow s'$
  end while
  ▷ *Compute Utilities*
  for each time-step t until the end of sampled episode **do**
    $\hat{\omega}_t$ gets $\hat{U}^i_{s_t}(\omega^{1:N}) = r(s_t, \hat{\omega}_t) + \gamma \cdot \max_k \mathbf{b}^k_{s_{t+1}} - \max_{j \neq i} \mathbf{b}^j_{s_t}$
    Losers ω^j_t get $U^j_{s_t}(\omega^{1:N}) = 0$
    for all primitives ω^i **do**
      Primitive ω^i_t stores $(s_t, \mathbf{b}^i_{s_t}, s_{t+1}, U^i_{s_t}(\omega^i_t))$ into m^i
    end for
  end for
  ▷ *Update*
  if time to update **then**
    for all primitives ω^i **do**
      Update primitive ω^i with update rule f with memory m^i
    end for
  end if
end while

E. Implementations of an On-Policy Decentralized Reinforcement Learning Algorithm

Stochastic policy gradient In this paper, we considered optimizing the bidding policies with a stochastic policy gradient algorithm, which means that the bidding policies parameterize a distribution of bids. The stochasticity of the bidding policy means that there need not be an explicit exploration strategy such as ϵ-greedy, which simplifies the analysis in this paper. In future work it would be interesting to explore deterministic bidding policies as well.

Bidding Policies We implement all bidding policies as neural networks that output the parameters of a Beta-distribution. Because the Beta-distribution has support between 0 and 1, we normalized environment rewards so returns fit in this range. In theory, the range of possible bids could be $[0, \infty)$ and need not be restricted to $[0, 1]$. The society decision-making framework prescribes a different unique local auction, with different primitives, at each state s. However, because the state space could be extremely large or even continuous, in practice we share the same set of primitives $\omega^{1:N}$ across all states, express their bidding policies as functions of the state, and rely on the function approximation capabilities of neural networks to learn different state-conditioned bidding strategies.

Redundancy Implementing two clones of each primitive means that each clone should share the same transformation ϕ: if ϕ were the literal action of `go-left`, then there would be two primitives that bid in the local auction to execute the `go-left` action. We implemented redundant clones by sharing the weights of their bidding policies ψ. Therefore, cloned primitives have identical bidding distributions, from which different bids are sampled. Alternatively, we also explored giving clone primitives the same transformation ϕ but independently parameterized bidding policies ψ. While we did not find much difference in global performance with independently parameterized bidding policies, such a scheme could be useful for multi-task learning where the same transformation could have different optimal societal Q-values depending on the task.

Learning Objectives The utility of the cloned Vickrey society is the learning objective. We compared three possible implementations of this learning objective: bucket-brigade (*BB*), Vickrey (*V*), and credit-conserving Vickrey (*CCV*) as described in Section 7.1.2 and Figure 4. We also compared with a baseline that sets the learning objective to be the environment reward. For all implementations and the baseline, the learning objective that a loser ω that the auction receives is 0 because its allocation $X^i(\mathbf{b})$ and payment $P^i(\mathbf{b})$ are both 0. Letting $\hat{\mathbf{b}}_{s_t}$ and $\mathbf{b}'_{s_t}$ denote the highest and second highest bid at time t respectively, the learning objective $\hat{U}^i_{s_t}(\omega^{1:N})$ of the winner $\hat{\omega}$ for state s_t is given below.

	Learning Objective
Bucket-Brigade	$\hat{U}^i_{s_t}(\omega^{1:N}) = \left[r(s_t, \hat{\omega}_t) + \gamma \cdot \hat{\mathbf{b}}_{s_{t+1}}\right] - \hat{\mathbf{b}}_{s_t}$
Vickrey	$\hat{U}^i_{s_t}(\omega^{1:N}) = \left[r(s_t, \hat{\omega}_t) + \gamma \cdot \hat{\mathbf{b}}_{s_{t+1}}\right] - \mathbf{b}'_{s_t}$
Credit-Conserving Vickrey	$\hat{U}^i_{s_t}(\omega^{1:N}) = \left[r(s_t, \hat{\omega}_t) + \gamma \cdot \mathbf{b}'_{s_{t+1}}\right] - \mathbf{b}'_{s_t}$
Environment Reward	$\hat{U}^i_{s_t}(\omega^{1:N}) = [r(s_t, \hat{\omega}_t)]$

F. Experimental Details

We implemented our experiments using the PyTorch library (Paszke et al., 2019). For all experiments, for proximal policy optimization we used a policy learning rate of $4 \cdot 10^{-5}$, a value function learning rate of $5 \cdot 10^{-3}$, a clipping ratio of 0.2, a GAE (Schulman et al., 2015) smoothing parameter of 0.95, a discount factor of 0.99, and the Adam (Kingma & Ba, 2014) optimizer. Each bidding policy has its own replay buffer. Each bidding policy is updated every 4096 transitions with a minibatch size of 256. For *Two Rooms* and *Mental Rotation* we added an entropy bonus, with a weighting coefficient of 0.1, to the PPO loss.

The plots for *Market Bandit*, *Two Rooms*, and *Mental Rotation* were averaged over 3 seeds and the other plots were averaged over 5 seeds. The metric for the learning curves is the mean return over 4096 environment steps. The error bars represent the 10th, 50th, 90th quantile. All experiments were run on CPU, except for the *TwoRooms* environment, which was run on GPU.

F.1. Numerical Simulations

All transformations ϕ correspond to literal actions. The bidding policies ψ are implemented as linear neural networks with a single hidden layer of 16 units that output the α and β parameters of the Beta distribution. There is no activation function for the hidden layer. We used the softplus activation function to output α and β. The valuation functions for the bidding policies are also implemented as linear neural networks with a single hidden layer of 16 units.

F.1.1. Market Bandit

At every round, we stochastically drop out a subset of primitives. For cloned societies, one clone could be dropped out while the other clone stays in the auction. The drop-out sampling procedure is as follows. Letting N be the number of total primitives (which means there are $N/2$ unique primitives in cloned societies), we first sample a random integer m in $\{2, 3, ..., N\}$. Then we sample a subset of m primitives from the N total primitives without replacement. For a given round, only the primitives that participated in that round are updated.

F.1.2. DUALITY

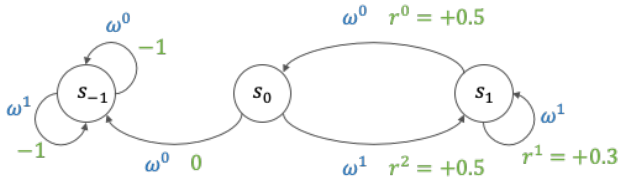

Figure 11. **The *Duality* environment.**

As stated in Figure 4, without redundant primitives the solitary *CCV* implementation sacrifices Bellman optimality in general. We now use the *Duality* environment as an example to illustrate such an instance where the dominant strategy of the primitives does not coincide with the global optimal policy of the society, even if the primitives have full knowledge of their own valuations. The DSIC property of the Vickrey auction makes it straightforward to analyze the dominant strategies of the primitives, which the following paragraphs illustrate. Recall that based on the *CCV* learning objective, the valuation of the winner $\hat{\omega}$ at time t is the immediate reward plus the discounted *second*-highest bid (not the highest bid) at the next time-step: $r(s_t, \hat{\omega}_t) + \gamma \cdot \mathbf{b}'_{s_{t+1}}$. Let r^0 be equal to the environment reward $r(s_1, \omega^0)$, r^1 be equal to the environment reward $r(s_1, \omega^1)$, and r^2 be equal to the environment reward $r(s_0, \omega^1)$, where in Figure 11 we see that $r^0 = 0.5$, $r^1 = 0.3$, and $r^2 = 0.5$

At state s_0, primitive ω^0 will bid 0, the lowest possible bid, because the local auction at state s_0 would lead to unbounded negative reward at s_{-1}. This means that the valuation that ω^0 has for winning at state s_1 is $r^0 + \gamma \cdot 0 = r(s_1, \omega^0)$, since 0 must be the second highest bid at state s_0. Thus the dominant strategy for ω^0 is to truthfully bid r^0. At s_1, primitive ω^1 will bid some number c. Since activating ω^1 is a self-loop, ω^1 can sell s_1 back to itself in the next time-step. Thus the valuation that ω^1 has for winning at state s_1 is $r^1 + \gamma \cdot \min(r^0, c)$. Thus the dominant strategy for ω^0 is to truthfully bid $c = r^1 + \gamma \cdot \min(r^0, c)$.

In the undiscounted case, where $\gamma = 1$, if we solve for c, we have that if $r^1 > 0$, then $c = r^1 + r^0$, which is greater than r^0, which is what ω^0 would bid as its dominant strategy. Therefore, in the case that $r^1 > 0$, ω^1 will continue to sell s_1 to itself. As long as $r^1 < r^0 + r^2$, the self-loop at s_1 will be less optimal than cycling back and forth between s_0 and s_1.

In the discounted case, where $0 < \gamma < 1$ and we again assume that $r^1 > 0$. Solving c again, we see that if in the case that $r^0 < \frac{r^1}{1-\gamma}$, then $c = r^1 + \gamma r^0$ and $c > r^0$. In this case ω^1 will bid higher than ω^0 at state s_1, creating a perpetual self-loop. If instead $r^0 > \frac{r^1}{1-\gamma}$, then $c = \frac{r^1}{1-\gamma}$ and $c < r^0$. In this case ω^0 will bid higher than ω^1 at state s_1, which is the optimal global policy.

Therefore the *Duality* environment illustrates that without redundancy, for fortuitous settings of r^0 and r^1, the dominant strategy equilibrium of the society may coincide with the optimal global policy, but if the for other choices of r^0 and r^1, the dominant strategy equilibrium of the society may be globally suboptimal. Adding redundant primitives makes the auction utilities consistent with the Bellman optimality equations and therefore does not suffer from such suboptimal equilibria.

F.2. Two Rooms

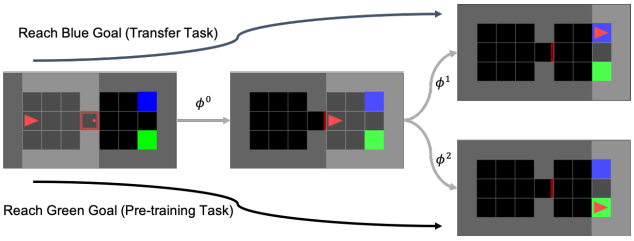

Figure 12. **The *Two Rooms* environment.**

The transformations $\phi^{0:2}$ are subpolicies pre-trained with PPO and have an action space equivalent to the action space of the Minigrid environment. For these transformation subpolicies, the bidding policies, the value functions for the bidding

policies, the non-hierarchical monolithic baseline, and the hierarchical monolithic baseline, we adapted the convolutional neural network architecture from https://github.com/lcswillems/rl-starter-files.

ϕ^0 is initialized randomly in the room on the left and is pre-trained to take the done action once it opens the red door and enters the room on the right, upon which it receives a terminal reward. ϕ^1 is initialized randomly in the room on the right and is pre-trained to take the done action upon reaching the green square, upon which it receives a terminal reward. ϕ^2 is initialized randomly in the room on the right and is pre-trained to take the done action upon reaching the blue square, upon which it receives a terminal reward.

In the pre-training task, the society receives a terminal reward upon reaching the green square and no intermediate rewards. In the transfer task, the society receives a terminal reward upon reaching the blue square and no intermediate rewards. These subpolicies $\phi^{0:2}$ are frozen for these tasks. For all learners, we trained to convergence on the pre-training task, then we initialized training for the transfer task from the best saved checkpoint on the pre-training task. The non-hierarchical monolithic baseline could not solve the pre-training task so we did not consider it for the transfer task.

F.3. Mental Rotation

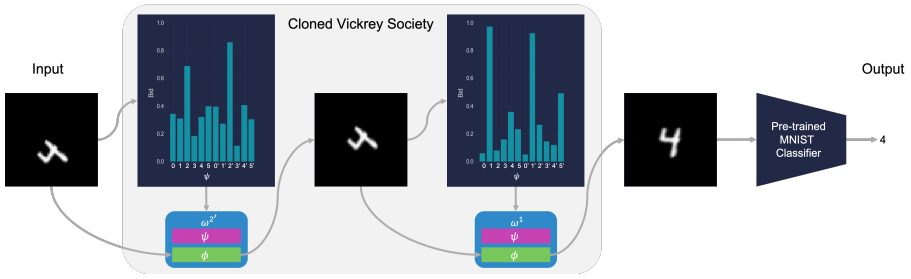

Figure 13. **The *Mental Rotation* environment.**

Environment Each 28×28 image in the MNIST training set was first inset into a black background of 64×64. Then the image first rotated either clockwise or counterclockwise by 60 degrees, then translated left, up, down, right by 29% of the image width, for a total of eight possible transformation combinations given these six affine transformations. These transformation combinations were taken from Chang et al. (2018).

Primitives The transformations ϕ correspond to the same six affine transformations, summarized below:

$\phi^0, \phi^{0'}$	`rotate-counterclockwise`
$\phi^1, \phi^{1'}$	`rotate-clockwise`
$\phi^2, \phi^{2'}$	`translate-up`
$\phi^3, \phi^{3'}$	`translate-down`
$\phi^4, \phi^{4'}$	`translate-left`
$\phi^5, \phi^{5'}$	`translate-right`

The primitive ω^i, and its bidding policy ψ^i, correspond to the transformation ϕ^i. The clones are indicate by i and i'.

Neural network architecture The pre-trained MNIST classifier, the bidding policies, and the value functions for the bidding policies follow the same architecture as the convolutional neural network used in Chang et al. (2018), with different output dimensions (10 as the output dimension of the MNIST classifier, 1 for the other networks). The PyTorch architecture is given below:

```
network = nn.Sequential(
            nn.Conv2d(1, 8, 4, 2, 1, bias=False),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(8, 8 * 2, 4, 2, 1, bias=False),
            nn.BatchNorm2d(8 * 2),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(8 * 2, 8 * 4, 4, 2, 1, bias=False),
            nn.BatchNorm2d(8 * 4),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(8 * 4, 8 * 8, 4, 2, 1, bias=False),
            nn.BatchNorm2d(8 * 8),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(8 * 8, outdim, 4, 1, 0, bias=False))
```