# Affordance Math

## 1 Affordances

Affordances are defined as a mapping from an OO-MDP state $s$ to a set of actions $\mathcal{A}$.

Each affordance also maintains a tuple $\langle p, g \rangle$, where:

- $p$ is a predicate on states, representing the precondition for activating the affordance
- $g$ is a goal type, representing the type of problem the agent is solving.

The mapping occurs by evaluating the precondition $p$ in state $s$, and determining if the goal type is currently relevant (direct equivalence to current goal, logical entailment of goal types, etc), and if so, returning an action set $\mathcal{A}$. The action set is either provided by a domain expert, or is learned and applied as follows.

## 2 Computing $\mathcal{A}$

$\mathcal{A}$ is the result of taking $n$ samples from a multinomial over actions:

$$\mathcal{A} \leftarrow_n Mult(A) \tag{1}$$

Where $A$ is the OO-MDP action set. This multinomial, and $n$, may be specified by a domain expert, forcing $\mathcal{A}$ to be deterministic if the expert chooses.

If an expert is not involved, then we need to learn two things:

1. The multinomial over actions to sample from
2. How many actions to sample from the multinomial

## 3 Learning

We define $\mathcal{A}^*$ to be the set of of optimal actions for each state $s$, and define $\mathcal{K}^*$ to be the action set suggested by the affordance knowledge base for state $s$.

Since each affordance $\Delta_i$ maps to a set of actions for a given state $s$, $\mathcal{K}^*$ is defined as:

$$\mathcal{K}^* = \Delta_1(s) \cup \ldots \cup \Delta_K(s) \tag{2}$$

**Our goal is to create the affordance knowledge base that, in each state, $\mathcal{A}^* = \mathcal{K}^*$**

$$\Pr(\mathcal{K}^* = \mathcal{A}^* | s, \Delta_1 \ldots \Delta_K) \tag{3}$$

More specifically:

1) Each action in $\mathcal{A}^*$ is in $\mathcal{K}^*$

2) Each action *not* in $\mathcal{A}^*$ is *not* in $\mathcal{K}^*$.

$$\Pr(\mathcal{K}^* = \mathcal{A}^* | s, \Delta_1 \ldots \Delta_K) = \left[ \prod_i^{|\mathcal{A}^*|} \sum_j^{|\Delta|} \Pr(a_i \in \Delta_j(s) \mid s, \Delta_j) \right] \cdot \left[ 1 - \prod_i^{|\bar{\mathcal{A}}^*|} \sum_j^{|\Delta|} \Pr(b_i \in \Delta_j(s) \mid s, \Delta_j) \right] \tag{4}$$

Where $a_i$ is the $i$-th action in $\mathcal{A}^*$, and $b_i$ is the $i$-th action in $\bar{\mathcal{A}}^*$ (the set of non-optimal actions).

Equation 4 represents the probability that each optimal action is in the action set contributed by the affordances, and that each non-optimal action is not in the action set contributed by the affordances.

Furthermore, we define the probability that the $i$-th action is in the given affordances action set to be:

$$\Pr(a_i \in \Delta_j(s) \mid s, \Delta_j) = DirMult(\Delta_j.\alpha, \Delta_j.n) \tag{5}$$

Where $\Delta_j.\alpha$ denotes the hyper parameter vector for the Dirichlet-multinomial, and $\Delta_j.n$ indicates how many samples to draw. We define $\Delta_j.n$ to be a sample from a Dirichlet distribution over the affordances hyper parameter vector $\beta$:

$$\Delta_j.n \sim Dir(\Delta_j.\beta) \tag{6}$$

Therefore, in order to solve Equation 3, we need supply the Dirichlet-multinomial hyper parameters $\Delta_j.\alpha$ and $\Delta_j.\beta$. These are precisely the parameters that are learned during training.

We learn these parameters by synthesizing the optimal policy in several smaller worlds that parallel the test world, but are small enough to be solved with tabular methods. From this policy, we receive the optimal action set, $\mathcal{B}^*$ for each state for each of those worlds.

Because of the state-space dependence of the OO-MDP, we make the assumption that the optimal actions in similar states in the training worlds will be similar to the optimal action set in similar states in the test world. For more information on how these parameters are computed, see the pseudocode of the algorithms in the paper.