Implicit value updating explains transitive inference performance: The betasort model (Supplemental Information)

Greg Jensen[1,2*], Fabian Muñoz[1], Yelda Alkan[1], Vincent P. Ferrera[1,3], Herb Terrace[2]

Algorithm 1: The betasort updating policy.

```
Data: memory arrays U, L, R, N, chosen stimulus ch, unchosen
      stimulus nc, outcome r, recall ξ
Result: updated model U, L, R, N
begin
    R ← R · ξ; N ← N · ξ                          /* Relax R and N */
    E ← R/(R + N)                  /* Estimate trial reward rates */
    ξ_R ← E/(E + 1) + 0.5
    U ← U · ξ_R · ξ; L ← L · ξ_R · ξ               /* Relax U and L */
    V ← U/(U + L)                  /* Estimate stimulus positions */
    if r = 1 then
        U ← U + V; L ← L + (1 − V)  /* consolidate all stimuli */
    else
        U_nc ← U_nc + 1                              /* shift nc up */
        L_ch ← L_ch + 1                            /* shift ch down */
        for j = 1 to 7 do
            if j ≠ ch AND j ≠ nc then
                if V_j > V_ch AND V_j < V_nc then
                    U_j ← U_j + V_j; L_j ← L_j + (1 − V_j)  /* consolidate j
                    */
                else if V_j < V_nc then
                    L_j ← L_j + 1                         /* shift j down */
                else if V_j > V_ch then
                    U_j ← U_j + 1                           /* shift j up */
    return U, L, R, N
```

Algorithm 2: The beta$Q$ updating policy.

Data: memory arrays $\mathbf{U}$, $\mathbf{L}$, $\mathbf{R}$, $\mathbf{N}$, chosen stimulus ch, unchosen stimulus nc, outcome r, recall ξ
Result: updated model $\mathbf{U}$, $\mathbf{L}$, $\mathbf{R}$, $\mathbf{N}$

```
begin
    R_ch ← R_ch · ξ; R_nc ← R_nc · ξ                    /* Relax R_ch and R_nc */
    N_ch ← N_ch · ξ; N_nc ← N_nc · ξ                    /* Relax N_ch and N_nc */
    E ← R/(R + N)                           /* Estimate trial reward rates */
    ξ_R ← E/(E + 1) + 0.5
    U_ch ← U_ch · ξ_R_ch · ξ; U_nc ← U_nc · ξ_R_nc · ξ  /* Relax U_ch and U_nc */
    L_ch ← L_ch · ξ_R_ch · ξ; L_nc ← L_nc · ξ_R_nc · ξ  /* Relax L_ch and L_nc */
    V ← U/(U + L)                           /* Estimate stimulus positions */
    if r = 1 then
        U_ch ← U_ch + V_ch; L_ch ← L_ch + (1 − V_ch)  /* consolidate ch */
        U_nc ← U_nc + V_nc; L_nc ← L_nc + (1 − V_nc)  /* consolidate nc */
    else
        U_nc ← U_nc + 1                                    /* shift nc up */
        L_ch ← L_ch + 1                                  /* shift ch down */
    return U, L, R, N
```

Algorithm 3: The Q-learning updating policy.

Data: memory array $\mathbf{Q}$, chosen stimulus ch, unchosen stimulus nc, outcome r, modifier α
Result: updated model $\mathbf{Q}$

```
begin
    if r = 1 then
        Q_ch ← Q_ch + α · (1 − Q_ch)                       /* shift ch up */
        Q_nc ← Q_nc − α · Q_nc                           /* shift nc down */
    else if r = 0 then
        Q_ch ← Q_ch − α · Q_ch                           /* shift ch down */
        Q_nc ← Q_nc + α · (1 − Q_nc)                       /* shift nc up */
    return Q
```