Algorithms to Compute Appraisal Variables

Algorithm 1 (Relevance)

function IsEventRelevant(*Event* ε_t)

    Initialize graph $\mathcal{G}_t$ with current mental state $\mathcal{S}_t$.

    $g_t \leftarrow$ extractGoal($\mathcal{G}_t$)

    $\mathcal{P}_t \leftarrow$ extractPaths(ε_t, g_t)

    if ($\mathcal{P}_t = \emptyset$) **then**
        return FALSE
    else
        $\mathcal{U}_t \leftarrow$ getEventUtility(ε_t, g_t)
        if ($\mathcal{U}_t \geq \tau_e$) **then**
            return TRUE
        else
            return FALSE
end function

Algorithm 2 (Desirability)

```
1: function IsEventDesirable(Event ε_t)

2:     Initialize graph G_t with current mental state S_t.

3:     if (topLevelTaskStatus() = ACHIEVED) then
4:         return TRUE
5:     else if (topLevelTaskStatus() = BLOCKED) then
6:         return FALSE
7:     else if (topLevelTaskStatus() = INPROGRESS) then

8:         if (currentTaskStatus() = ACHIEVED) then
9:             return TRUE
10:        else if (currentTaskStatus() = BLOCKED) then
11:            return FALSE
12:        else if (currentTaskStatus() = INPROGRESS) then
13:            return TRUE
14:        else if (currentTaskStatus() = UNKNOWN) then

15:            if (taskPreconditionStatus() = SATISFIED) then
16:                return TRUE
17:            else if (taskPreconditionStatus() = UNSATISFIED) then
18:                return FALSE
19:            else if (taskPreconditionStatus() = UNKNOWN) then

20:                if (doesContribute(ε_t, g⃗_t) = TRUE) then
21:                    return TRUE
22:                else if (doesContribute(ε_t, g⃗_t) = FALSE) then

23:                    g_t ← extractGoals(G_t)
24:                    U_t ← getEventUtility(ε_t, g_t)
25:                    if (U_t ≥ τ_e) then
26:                        return TRUE
27:                    else
28:                        return FALSE
29:                    end if

30:                end if
31:            end if
32:        end if
33:    end if
34: end function
```

Algorithm 3 (Expectedness)

function IsEventExpected(*Event* ε_t)

    Initialize graph $\mathcal{G}_{t-1}$ with previous mental state $\mathcal{S}_{t-1}$.
    Initialize graph $\mathcal{G}_t$ with current mental state $\mathcal{S}_t$.

    $g_{t-1} \leftarrow$ extractGoals($\mathcal{G}_{t-1}$)
    $g_t \leftarrow$ extractGoals($\mathcal{G}_t$)

    if ($g_t \neq g_{t-1}$) **then**
        if (IsAchieved(g_{t-1}) = FALSE) **then**
            return FALSE
        else
            $\mathcal{P}_t \leftarrow$ extractPaths(ε_t, g_t)
            if ($\mathcal{P}_t = \emptyset$) **then**
                return FALSE
            else
                $\mathcal{U}_t \leftarrow$ getPathUtility($\mathcal{G}_t, g_t$)
                if ($\mathcal{U}_t \geq \tau_e$) **then**
                    return TRUE
                else
                    return FALSE
    else
        $\mathcal{U}_t \leftarrow$ getPathUtility($\mathcal{G}_t, g_t$)
        $\mathcal{U}_{t-1} \leftarrow$ getPathUtility($\mathcal{G}_t, g_{t-1}$)
        if ($(\mathcal{U}_t - \mathcal{U}_{t-1}) \geq \tau_e$) **then**
            return TRUE
        else
            return FALSE

end function

Algorithm 4 (Controllability)

function IsEventControllable(*Event* ε_t)

Initialize graph $\mathcal{G}_t$ with current mental state $\mathcal{S}_t$.

$g_t \leftarrow$ extractGoals($\mathcal{G}_t$)
$\mathcal{P}_{g_t} \leftarrow$ extractPaths(ε_t, g_t)

$\alpha^{agency}_{self/other} \leftarrow$ GetAgencyRatio(g_t)

$\beta^{autonomy}_{self/other} \leftarrow$ GetAutonomyRatio(g_t)

$\lambda^{predecessors}_{succeeded/total} \leftarrow$ GetSucceededPredecessorsRatio(g_t)

$\mu^{inputs}_{available/required} \leftarrow$ GetAvailableInputRatio(g_t)

$\mathcal{U}_t \leftarrow \frac{\omega_0 \cdot \alpha^{agency}_{self/other} + \omega_1 \cdot \beta^{autonomy}_{self/other} + \omega_2 \cdot \lambda^{predecessors}_{succeeded/total} + \omega_3 \cdot \mu^{inputs}_{available/required}}{\omega_0 + \omega_1 + \omega_2 + \omega_3}$

if $(\mathcal{U}_t \geq \tau_e)$ **then**
    return TRUE
else
    return FALSE
end function

Algorithm 5 (Get Succeeded Predecessors Ratio)

function GETSUCCEEDEDPREDECESSORSRATIO(*Goal* g_t)

  $\Phi_g \leftarrow$ EXTRACTPREDECESSORS(g_t)

  for each $\phi_g^i \in \Phi_g$ **do**
    if (ISACHIEVED(ϕ_g^i)) **then**
      $count_{predecessor}^{achieved} \leftarrow count_{predecessor}^{achieved} + 1$
    $count_{predecessor}^{total} \leftarrow count_{predecessor}^{total} + 1$

  return $\frac{count_{predecessor}^{achieved}}{count_{predecessor}^{total}}$

end function

Algorithm 6 (Get Available Input Ratio)

function GETAVAILABLEINPUTRATIO(*Paths* $\mathcal{P}_{\tilde{g}}^A$)

  $\mathcal{X}_g \leftarrow$ EXTRACTINPUTS($\mathcal{P}_g^A$)

  for each $\chi_g^i \in \mathcal{X}_g$ **do**
    if (ISAVAILABLE(χ_g^i)) **then**
      $count_{input}^{available} \leftarrow count_{input}^{available} + 1$
    $count_{input}^{required} \leftarrow count_{input}^{required} + 1$

  return $\frac{count_{input}^{available}}{count_{input}^{required}}$

end function

Algorithm 7 (Get Agency Ratio)

function GETAGENCYRATIO(*Paths* $\mathcal{P}^{A}_{\vec{g}}$)
    $count^{self}_{responsibility} \leftarrow count^{other}_{responsibility} \leftarrow 0$
    $\Theta_{\vec{g}} \leftarrow$ EXTRACTPRECONDITIONS($\mathcal{P}^{A}_{\vec{g}}$)
    for each $\theta^{i}_{\vec{g}} \in \Theta_{\vec{g}}$ **do**
        if (GETRESPONSIBLE($\theta^{i}_{\vec{g}}$) = SELF) **then**
            $count^{self}_{responsibility} \leftarrow count^{self}_{responsibility} + 1$
        else
            $count^{other}_{responsibility} \leftarrow count^{other}_{responsibility} + 1$
    return $\langle count^{self}_{responsibility}, count^{other}_{responsibility} \rangle$
end function

Algorithm 8 (Get Autonomy Ratio)

function GETAUTONOMYRATIO(*Paths* $\mathcal{P}^{A}_{\vec{g}}$)
    $\mathcal{A} \leftarrow$ EXTRACTACTION($\mathcal{P}^{A}_{\vec{g}}$)
    $\mathcal{R}_{\mathcal{A}} \leftarrow$ GETRESPONSIBLE($\mathcal{A}$)
    $\mathcal{M}_{\mathcal{R}_{\mathcal{A}}} \leftarrow$ GETMOTIVE($\mathcal{R}_{\mathcal{A}}$)
    if ($\mathcal{M}_{\mathcal{R}_{\mathcal{A}}} \neq \emptyset$) **then**
        return MAX
    else
        return MIN
end function