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 0
  else
    $\mathcal{U}_t \leftarrow$ getEventUtility(ε_t, g_t)
    if ($\mathcal{U}_t \geq \tau_e$) **then**
      return ($\mathcal{U}_t$)
    else
      return 0
end function

Algorithm 2 (Desirability)

function IsEventDesirable(*Event* ε_t)

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

$\vec{g}_t \leftarrow$ extractGoals($\mathcal{G}_t$)

```
if (topLevelTaskStatus() = ACHIEVED) then
    return 1.0
else if (topLevelTaskStatus() = BLOCKED) then
    return -1.0
else if (topLevelTaskStatus() = INPROGRESS) then
    if (currentTaskStatus() = ACHIEVED) then
        return 0.75
    else if (currentTaskStatus() = BLOCKED) then
        return -0.75
    else if (currentTaskStatus() = INPROGRESS) then
        return 0.25
    else if (currentTaskStatus() = UNKNOWN) then
        if (taskPreconditionStatus() = SATISFIED) then
            return 0.5
        else if (taskPreconditionStatus() = UNSATISFIED) then
            return -0.75
        else if (taskPreconditionStatus() = UNKNOWN) then
            if (doesContribute(ε_t, g⃗_t) = TRUE) then
                return 0.0
            else if (doesContribute(ε_t, g⃗_t) = FALSE) then
                if (recipeApplicability(ε_t, g⃗_t) = APPLICABLE) then
                    return -0.5
                else if (recipeApplicability(ε_t, g⃗_t) = INAPPLICABLE) then
                    return -0.75
                else if (recipeApplicability(ε_t, g⃗_t) = UNKNOWN) then
                    return -0.25
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)

$\alpha^{agency}_{self/other} \leftarrow \beta^{autonomy}_{self/other} \leftarrow 0$

$\lambda^{predecessors}_{succeeded/failed} \leftarrow \mu^{inputs}_{available/required} \leftarrow 0$

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

$\vec{g}_t \leftarrow \textsc{extractGoals}(\mathcal{G}_t)$

$\mathcal{P}_{\vec{g}_t} \leftarrow \textsc{extractPaths}(\varepsilon_t, \vec{g}_t)$

$\alpha^{agency}_{self/other} \leftarrow \textsc{GetAgencyValue}(\mathcal{P}_{\vec{g}_t})$

$\beta^{autonomy}_{self/other} \leftarrow \textsc{GetAutonomyValue}(\mathcal{P}_{\vec{g}_t})$

$\lambda^{predecessors}_{succeeded/total} \leftarrow \textsc{GetSucceededPredecessorsRatio}(\mathcal{P}_{\vec{g}_t})$

$\mu^{inputs}_{available/required} \leftarrow \textsc{GetAvailableInputRatio}(\mathcal{P}_{\vec{g}_t})$

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

if $(\mathcal{U}_t \geq \tau_e)$ **then**

  return TRUE

else

  return FALSE

end function

Algorithm 5 (Check Predecessors)

function GetSucceededPredecessorsRatio(*Paths* $\mathcal{P}_{\vec{g}}^{A}$)

$count_{predecessor}^{succeeded} \leftarrow count_{predecessor}^{total} \leftarrow 0$

$\Phi_{\vec{g}} \leftarrow$ extractPredecessors($\mathcal{P}_{\vec{g}}^{A}$)

for each $\phi_{\vec{g}}^{i} \in \Phi_{\vec{g}}$ **do**

    if (IsSucceeded($\phi_{\vec{g}}^{i}$)) **then**

        $count_{predecessor}^{succeeded} \leftarrow count_{predecessor}^{succeeded} + 1$

    $count_{predecessor}^{total} \leftarrow count_{predecessor}^{total} + 1$

return $\langle count_{predecessor}^{succeeded}, count_{predecessor}^{total} \rangle$

end function

Algorithm 6 (Check Inputs)

function GetAvailableInputRatio(*Paths* $\mathcal{P}_{\vec{g}}^{A}$)

$count_{input}^{available} \leftarrow count_{input}^{required} \leftarrow 0$

$\mathcal{X}_{\vec{g}} \leftarrow$ extractInputs($\mathcal{P}_{\vec{g}}^{A}$)

for each $\chi_{\vec{g}}^{i} \in \mathcal{X}_{\vec{g}}$ **do**

    if (IsSucceeded($\chi_{\vec{g}}^{i}$)) **then**

        $count_{input}^{available} \leftarrow count_{input}^{available} + 1$

    $count_{input}^{required} \leftarrow count_{input}^{required} + 1$

return $\langle count_{input}^{available}, count_{input}^{required} \rangle$

end function

Algorithm 7 (Get Agency Value)

```
function GetAgencyValue(Paths P_g^A)
    count_responsibility^self ← count_responsibility^other ← 0
    Θ_g ← ExtractPreconditions(P_g^A)
    for each θ_g^i ∈ Θ_g do
        if (GetResponsible(θ_g^i) = SELF) then
            count_responsibility^self ← count_responsibility^self + 1
        else
            count_responsibility^other ← count_responsibility^other + 1
    return ⟨count_responsibility^self, count_responsibility^other⟩
end function
```

Algorithm 8 (Get Autonomy Value)

```
function GetAutonomyValue(Paths P_g^A)
    A ← ExtractAction(P_g^A)
    R_A ← GetResponsible(A)
    M_R_A ← GetMotive(R_A)
    if (M_R_A ≠ ∅) then
        return MAX
    else
        return MIN
end function
```