Algorithms to Compute Appraisal Variables

Algorithm 1 (Relevance)

1: **function** IsEventRelevant(*Events* ε_t)
2: Initialize graph $\mathcal{G}_t$ with current mental state $\mathcal{S}_t$.
3: $\vec{g}_t \leftarrow$ extractGoal($\mathcal{G}_t$)
4: $\mathcal{P}_t \leftarrow$ extractPaths($\varepsilon_t, \vec{g}_t$)
5: **if** ($\mathcal{P}_t = \emptyset$) **then**
6: **return** 0
7: **else**
8: $\mathcal{U}_t \leftarrow$ getEventUtility($\varepsilon_t, \vec{g}_t$)
9: **if** ($\mathcal{U}_t \geq \tau_e$) **then**
10: **return** ($\mathcal{U}_t$)
11: **else**
12: **return** 0
13: **end function**

Algorithm 2 (Desirability)

function IsEventDesirable(*Events* 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 ($doesContribute(\varepsilon_t, \vec{g}_t)$ = TRUE) **then**
                    return -0.5
                else if ($doesContribute(\varepsilon_t, \vec{g}_t)$ = FALSE) **then**
                    if ($recipeApplicability(\varepsilon_t, \vec{g}_t)$ = APPLICABLE) **then**
                        return -0.5
                    else if ($recipeApplicability(\varepsilon_t, \vec{g}_t)$ = INAPPLICABLE) **then**
                        return -0.75
                    else if ($recipeApplicability(\varepsilon_t, \vec{g}_t)$ = UNKNOWN) **then**
                        return -0.25
end function

Algorithm 3 (Expectedness)

function IsEventExpected(*Events* 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 $\mathcal{S}_t$.

  $\vec{g}_{t-1} \leftarrow$ extractGoals($\mathcal{G}_{t-1}$)
  $\vec{g}_t \leftarrow$ extractGoals($\mathcal{G}_t$)

  if ($\vec{g}_t \neq \vec{g}_{t-1}$) **then**
    if (IsAchieved($\vec{g}_{t-1}$)) **then**
      $\mathcal{P}_t \leftarrow$ extractPaths($\varepsilon_t, \vec{g}_t$)
      if ($\mathcal{P}_t = \emptyset$) **then**
        return FALSE
      else
        $\mathcal{U}_t \leftarrow$ getPathUtility($\mathcal{G}_t, \vec{g}_t$)
        $\mathcal{U}_{t-1} \leftarrow$ getPathUtility($\mathcal{G}_t, \vec{g}_{t-1}$)
        if ($\mathcal{U}_t - \mathcal{U}_{t-1}$) $\geq \tau_e$ **then**
          return TRUE
        else
          return FALSE
    else ▷ Collaboration goal has achieved.
      return TRUE
  else ▷ Goals have not changed.
    return TRUE
end function

Algorithm 4 (Controllability)

function IsEventControllable(*MentalStates* $\mathcal{S}_t$, *Event* ε_t)

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

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

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

$\vec{g}_t \leftarrow$ extractGoals($\mathcal{G}_t$)
$\mathcal{P}_{\vec{g}_t} \leftarrow$ extractPaths($\mathcal{G}_t, \vec{g}_t$)

$\mathcal{P}^A_{\vec{g}_t} \leftarrow$ ExtractActionPaths($\mathcal{P}_{\vec{g}_t}$)
$\varphi^A_{\vec{g}_t} \leftarrow$ GetRecipeApplicability($\mathcal{P}^A_{\vec{g}_t}$)

$\mathcal{P}^{\mathcal{E}}_{\vec{g}_t} \leftarrow$ ExtractEmotionPaths($\mathcal{P}_{\vec{g}_t}$)

$\mathcal{E}^{other}_t \leftarrow$ ExtractOthersEmotion($\mathcal{P}^{\mathcal{E}}_{\vec{g}_t}$)

$\mathcal{I}^{emotion}_{other} \leftarrow$ GetEmotionIntensity($\mathcal{E}^{other}_t$)

$\Upsilon_t \leftarrow$ GetSomaticMarkers($\mathcal{E}^{other}_t$)
$\mathcal{V}^{\mathcal{E}}_t \leftarrow$ GetEmotionValue($\Upsilon_t, \varepsilon_t$)

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

$\beta^{autonomy}_{self/other} \leftarrow {}^{\text{GetAutonomyValue}(\mathcal{P}_{g_t^{self}})}/_{\text{GetAutonomyValue}(\mathcal{P}_{g_t^{other}})}$

$Precond_{\mathcal{P}^A_{\vec{g}_t}} \leftarrow$ CheckPreconditions($\mathcal{P}^A_{\vec{g}_t}$)

$\kappa^{preconditions}_{satisfied/total} \leftarrow {}^{Precond^{satisfied}_{\mathcal{P}^A_{\vec{g}_t}}}/_{Precond^{total}_{\mathcal{P}^A_{\vec{g}_t}}}$

$Predecessors_{\mathcal{P}^A_{\vec{g}_t}} \leftarrow$ CheckPredecessors($\mathcal{P}^A_{\vec{g}_t}$)

$\lambda^{predecessors}_{succeeded/total} \leftarrow {}^{Predecessors^{succeeded}_{\mathcal{P}^A_{\vec{g}_t}}}/_{Predecessors^{total}_{\mathcal{P}^A_{\vec{g}_t}}}$

$Inputs_{\mathcal{P}^A_{\vec{g}_t}} \leftarrow$ CheckInputs($\mathcal{P}^A_{\vec{g}_t}$)

$\mu^{inputs}_{available/required} \leftarrow {}^{Inputs^{available}_{\mathcal{P}^A_{\vec{g}_t}}}/_{Inputs^{required}_{\mathcal{P}^A_{\vec{g}_t}}}$

$\mathcal{U}_t \leftarrow \frac{\omega_0 \cdot \alpha^{agency}_{self/other} + \omega_1 \cdot \beta^{autonomy}_{self/other}}{\omega_0 \cdot + \omega_1 \cdot}$

$+ \frac{\omega_2 \cdot \kappa^{preconditions}_{satisfied/total} + \omega_3 \cdot \lambda^{predecessors}_{succeeded/total} + \omega_4 \cdot \mu^{inputs}_{available/required}}{\omega_2 + \omega_3 + \omega_4}$

$+ \frac{\omega_5 \cdot \mathcal{I}^{emotion}_{other} + \omega_6 \cdot \varphi^{A}_{\vec{g}_t} + \omega_7 \cdot \mathcal{V}^{\mathcal{E}}_t}{\omega_5 + \omega_6 + \omega_7}$

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

Algorithm 5 (Check Predecessors)

function CHECKPREDECESSORS(*Paths* $\mathcal{P}^A_{\vec{g}}$)

$count^{succeeded}_{predecessor} \leftarrow count^{total}_{predecessor} \leftarrow 0$

$\Phi_{\vec{g}} \leftarrow$ EXTRACTPREDECESSORS($\mathcal{P}^A_{\vec{g}}$)

for each $\phi^i_{\vec{g}} \in \Phi_{\vec{g}}$ **do**
  if (ISSUCCEEDED($\phi^i_{\vec{g}}$)) **then**
    $count^{succeeded}_{predecessor} \leftarrow count^{succeeded}_{predecessor} + 1$
  $count^{total}_{predecessor} \leftarrow count^{total}_{predecessor} + 1$

return $\langle count^{succeeded}_{predecessor}, count^{total}_{predecessor} \rangle$
end function

Algorithm 6 (Check Inputs)

function CHECKINPUTS(*Paths* $\mathcal{P}^A_{\vec{g}}$)

$count^{available}_{input} \leftarrow count^{required}_{input} \leftarrow 0$

$\mathcal{X}_{\vec{g}} \leftarrow$ EXTRACTINPUTS($\mathcal{P}^A_{\vec{g}}$)

for each $\chi^i_{\vec{g}} \in \mathcal{X}_{\vec{g}}$ **do**
  if (ISSUCCEEDED($\chi^i_{\vec{g}}$)) **then**
    $count^{available}_{input} \leftarrow count^{available}_{input} + 1$
  $count^{required}_{input} \leftarrow count^{required}_{input} + 1$

return $\langle count^{available}_{input}, count^{required}_{input} \rangle$
end function

Algorithm 7 (Get Agency Value)

function GETAGENCYVALUE(*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 Value)

function GETAUTONOMYVALUE(*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