

Definição 1

Existe $E = \{e_1, \dots, e_n\}$, que corresponde as entidades. Existe dois tipos de entidades, Agentes representados pelo conjunto Ag e Artefatos representados pelo conjunto At onde $(E \equiv Ag \cup At) \wedge (ag_i \in Ag \rightarrow ag_i \notin At) \wedge (at_i \in At \rightarrow at_i \notin Ag)$

Definição 2

Existe relações de entidade para entidade que são mapeadas por um conjunto R . Isso é dado por meio da seguinte relação, $isRelation(r_k, e_i, e_j)$

Definição 3

ρ_{r_i} corresponde ao papel *receive* ou seja, o papel que recebe a atribuição, de um papel ρ_a , um papel *assigned* atribuidor, $R_r = \{\rho_{r_n}, \dots, \rho_{r_m}\} \wedge R_a = \{\rho_{a_n}, \dots, \rho_{a_k}\}$, $(R \equiv R_a \cup R_r) \wedge (\rho_m \in R_m \rightarrow \rho_m \notin R_a) \wedge (\rho_a \in R_a \rightarrow \rho_a \notin R_m)$.

Definição 4

Existe um conjunto D de relações deonticas. Existe $D_{permission}$ que é equivalente a D e existe $D_{obligation}$ que é um subconjunto de $D_{permission}$.

Definição 5

Existe um conjunto G de Objetivos. Existe dois tipos de objetivos, Ge e Gd , onde ambos são disjuntos. Ge contem os objetivos referentes ρ_r , ou seja - referente a quem é delegado para alguma função. Os objetivos de Gd são referentes a quem delega ρ_a . Agentes ρ_a de g_{d_k} tem a finalidade de atribuir ρ_m a determinado agente ag .

Definição 6

Existe um subconjunto de Ge conhecido como Gr , são objetivos de execução que podem ser afetado por eventos aleatórios com a finalidade de ocasionar um acidente. Esses eventos aleatórios possuem uma possibilidade de acontecer. Os agentes não têm controle sobre esses eventos.

Definição 7

$$attribution(ag, \rho_{r_i}) \wedge deonticRelation(\rho_r, g_{e_i}, d_{o_i}) \rightarrow obligation(ag, g_{e_i}) \quad (1)$$

Definição 8

$$attribution(ag, \rho_{r_i}) \wedge deonticRelation(\rho_{r_i}, g_{e_i}, d_{p_i}) \rightarrow permission(ag, g_{e_i}) \quad (2)$$

Definição 9

$$obligation(ag, g_{e_i}) \rightarrow permission(ag, g_{e_i}) \quad (3)$$

Definição 10

Se para iniciar um determinado objetivo g_{e_j} é necessário completar outros objetivos, g_{e_i}, \dots, g_{e_k} , então se pelo menos um destes objetivos for falso para

$isCompleted(g_x), x = i, \dots, k$, então $preCondition(g_j, (g_i, \dots, g_k))$ é falso, se não, então $preCondition(g_j, (g_i, \dots, g_k))$ é verdadeiro.

Sendo que existe um conjunto $g_{preRequisit} = \{g_i, \dots, g_k\}$, então

$$(preCondition(g_j, g_{preRequisite}) \rightarrow preRequisit(g_j)) \wedge \quad (4)$$

$$\neg(preCondition(g_j, g_{preRequisite}) \rightarrow \neg preRequisit(g_j)) \quad (5)$$

Definição 11

Sendo cm conjunto de condições que devem ser mantidos o tempo inteiro para que a atividade possa acontecer,

$$preRequisit(g_{e_i}) \wedge hasMaintainer(g_{e_i}, cm_i) \wedge isOk(cm_i) \rightarrow canStart(g_{e_i}) \quad (6)$$

Definição 12

$$preRequisit(g_{e_i}) \wedge hasMaintainer(g_{e_i}, cm_i) \wedge \neg isOk(cm_i) \wedge tryReach(g_{e_i}, ag) \rightarrow \quad (7)$$

$$violation(g_{e_i}, ag, cm_i) \quad (8)$$

Definição 13

Se $attribution(ag, \rho_r) \wedge \neg deonticRelation(\rho_r, g_{e_i}, d_{p_i}) \wedge canStart(g_{e_i}) \wedge tryReach(g_{e_i}, ag)$ é verdade, então $violation(d_{p_i}, ag)$ é verdade. Contudo, se $attribution(ag, \rho_r) \wedge \neg deonticRelation(\rho_r, g_e, d_{p_i}) \wedge canStart(g_e) \wedge tryReach(g_{e_i}, ag)$ for falso, então $violation(d_{p_i}, ag)$.

Definição 14

Existe c_r , onde c_r é um conjunto que mapeia as relações das entidades da pela relação $isRelationOf(r_j, c_{r_i})$. Cada g_{e_i} possui um c_{r_i} dada pela relação $hasConditionRelation(g_{e_i}, c_{r_i})$. Se $permission(ag, g_{e_i}) \wedge canStart(g_{e_i}) \wedge hasConditionRelation(g_{e_i}, c_{r_i}) \wedge isRelationOf(r_j, c_{r_i}) \wedge \neg know(ag, r_j) \wedge tryReach(ag, g_{e_i})$ é verdade, então é verdade $violation(r_j, g_{e_i}, ag)$. Contudo $violation(r_j, g_{e_i}, ag)$ é falso para a condição de $permission(ag, g_{e_i}) \wedge canStart(g_{e_i}) \wedge hasConditionRelation(g_{e_i}, c_{r_i}) \wedge isRelationOf(r_j, c_{r_i}) \wedge \neg know(ag, r_j) \wedge tryReach(ag, g_{e_i})$ ser falsa.

Definição 15

Se $permission(ag, g_{e_i}) \wedge canStart(g_{e_i}) \wedge hasConditionRelation(g_{e_i}, c_{r_i}) \wedge isRelationOf(r_j, c_{r_i}) \wedge \neg execute(ag, r_j) \wedge tryReach(ag, g_{e_i})$ é verdade, então é verdade $violation(r_j, g_{e_i}, ag)$. Contudo $violation(r_j, g_{e_i}, ag)$ é falso para a condição de $permission(ag, g_{e_i}) \wedge canStart(g_{e_i}) \wedge hasConditionRelation(g_{e_i}, c_{r_i}) \wedge isRelationOf(r_j, c_{r_i}) \wedge \neg execute(ag, r_j) \wedge tryReach(ag, g_{e_i})$ ser falsa.

Definição 16

Se $attribution(ag, \rho_r) \wedge deonticRelation(\rho_r, g_{e_i}, d_{o_i}) \wedge canStart(g_{e_i}) \wedge \neg tryReach(ag, g_{e_i})$ então existe uma violação dada como $violation(d_{o_i}, ag, g_{e_i})$. Se não $violation(d_{o_i}, ag, g_{e_i})$ é falso.

Definição 17

Existe um conjunto c_e , condition entity, que mapeia as entidades necessárias $isEntityOf(e_k, c_{e_i})$ para que um determinado objetivo possa ser alcançado. Para cada g_{e_i} existe uma dada relação em que $hasConditionEntity(g_{e_i}, c_{e_i})$. Sendo assim, $permission(ag, g_{e_i}) \wedge tryReach(ag, g_{e_i}) \wedge canStart(g_{e_i}) \wedge hasConditionEntity(g_{e_i}, c_{e_i}) \wedge isEntityOf(e_k, c_{e_i}) \wedge \neg isInMomentOfGoal(e_k, g_{e_i}) \rightarrow violation(g_{e_i}, e_k, ag)$. Contudo, é verdadeiro $\neg violation(g_{e_i}, e_k, ag)$ se $hasConditionEntity(g_{e_i}, c_{e_i}) \wedge tryReach(ag, g_{e_i}) \wedge canStart(g_{e_i}) \wedge hasConditionEntity(g_{e_i}, c_{e_i}) \wedge isEntityOf(e_k, c_{e_i}) \wedge \neg isInMomentOfGoal(e_k, g_{e_i})$ é falso. A relação $isInMomentOfGoal(e_k, g_{e_i})$ indica se entidade estará disponível no momento que o agente ag tenta alcançar o objetivo g_{e_i} .

Definição 18

$$attribution(ag, \rho_a) \rightarrow obligation(ag, g_d) \quad (9)$$

Definição 19

Se for verdade $obligation(ag_i, g_d) \wedge delegate(ag_i, \rho_{r_i}, ag_j) \wedge deonticRelation(\rho_{r_i}, g_{e_i}, d_p) \wedge hasConditionRelation(g_{e_i}, c_{r_i}) \wedge isRelation(r_k, c_{r_i}) \wedge \neg know(ag_j, r_k)$ então ocorre uma violação em $violation(\rho_a, g_d, ag_i)$. Se não, $\neg violation(\rho_a, g_d, ag_i)$ é verdade.

Definição 20

$$violation(d_{p_i}, ag) \rightarrow stopOperation \quad (10)$$

$$\neg violation(d_{p_i}, ag) \rightarrow \neg stopOperation \quad (11)$$

Definição 21

$$violation(r_j, g_{e_i}, ag) \wedge isGoalRandon(g_{e_j}) \wedge hasPossibilityBadEvent(g_{e_j}, p_1) \quad (12)$$

$$\rightarrow hasPossibilityBadEvent(g_{e_j}, p_2) \wedge p_2 > p_1 \quad (13)$$

$$\neg violation(r_j, g_{e_i}, ag) \wedge isGoalRandon(g_{e_j}) \wedge hasPossibilityBadEvent(g_{e_j}, p_1) \quad (14)$$

$$\rightarrow hasPossibilityBadEvent(g_{e_j}, p_1) \quad (15)$$

Definição 22

$$violation(r_j, g_{e_i}, ag) \rightarrow sanction(g_{e_i}, ag, risk) \rightarrow badEvent(risk, fatality, ag, g_{e_i}) \quad (16)$$

$$\neg violation(r_j, g_{e_i}, ag) \rightarrow \neg sanction(g_{e_i}, ag, risk) \rightarrow \neg badEvent(risk, fatality, ag, g_{e_i}) \quad (17)$$

Definição 23

$$hasPossibilityBadEvent(g_{e_j}, p) \wedge eventBadHappens(g_{e_j}) \wedge tryReach(ag, g_{e_j}) (18)$$

$$\rightarrow badEvent(risk, fatality, ag, g_{e_j}) (19)$$

$$hasPossibilityBadEvent(g_{e_j}, p) \wedge \neg eventBadHappens(g_{e_j}) \wedge tryReach(ag, g_{e_j}) (20)$$

$$\rightarrow \neg badEvent(risk, fatality, ag, g_{e_j}) (21)$$

$$hasPossibilityBadEvent(g_{e_j}, p) \wedge eventBadHappens(g_{e_j}) \wedge \neg tryReach(ag, g_{e_j}) (22)$$

$$\rightarrow \neg badEvent(risk, fatality, ag, g_{e_j}) (23)$$

Definição 24

$$violation(d_{o_i}, ag, g_{e_i}) \rightarrow stopOperation (24)$$

$$\neg violation(d_{o_i}, ag, g_{e_i}) \rightarrow \neg stopOperation (25)$$

Definição 25

$$violation(g_{e_i}, e_k, ag) \rightarrow sanction(g_{e_i}, ag, risk) \rightarrow badEvent(risk, fatality, ag, g_{e_i}) (26)$$

$$\neg violation(g_{e_i}, e_k, ag) \rightarrow \neg sanction(g_{e_i}, ag, risk) \rightarrow \neg badEvent(risk, fatality, ag, g_{e_i}) (27)$$

Definição 26

$$violation(g_{e_i}, ag, cm_i) \rightarrow sanction(g_{e_i}, ag, risk) \rightarrow badEvent(risk, fatality, ag, g_{e_i}) (28)$$

$$\neg violation(g_{e_i}, e_k, ag) \rightarrow \neg sanction(g_{e_i}, ag, risk) \rightarrow \neg badEvent(risk, fatality, ag, g_{e_i}) (29)$$

Definição 27

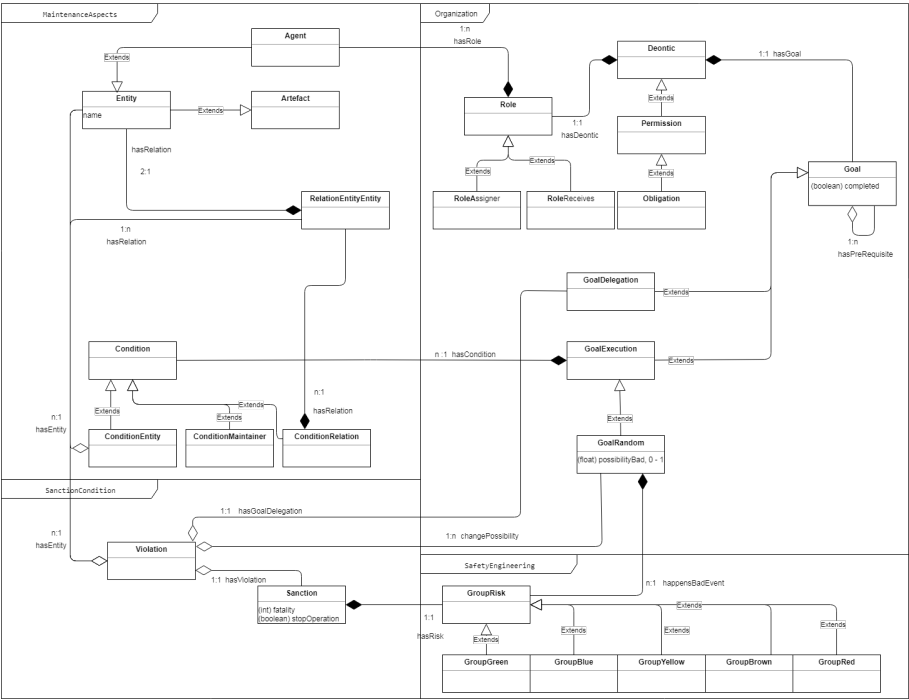
$$badEvent(risk, fatality, ag, g_{e_i}) \rightarrow stopOperation (30)$$

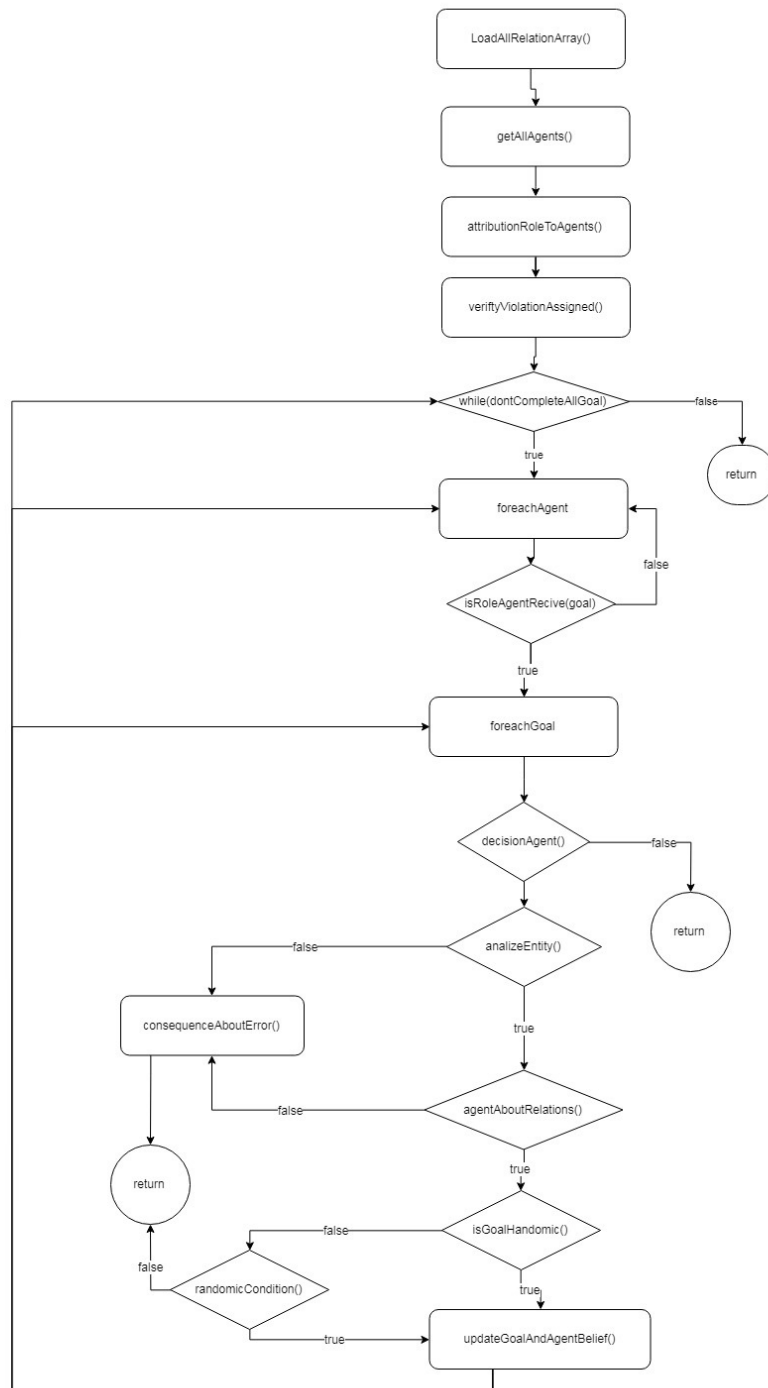
$$\neg badEvent(risk, fatality, ag, g_{e_i}) \rightarrow \neg stopOperation (31)$$

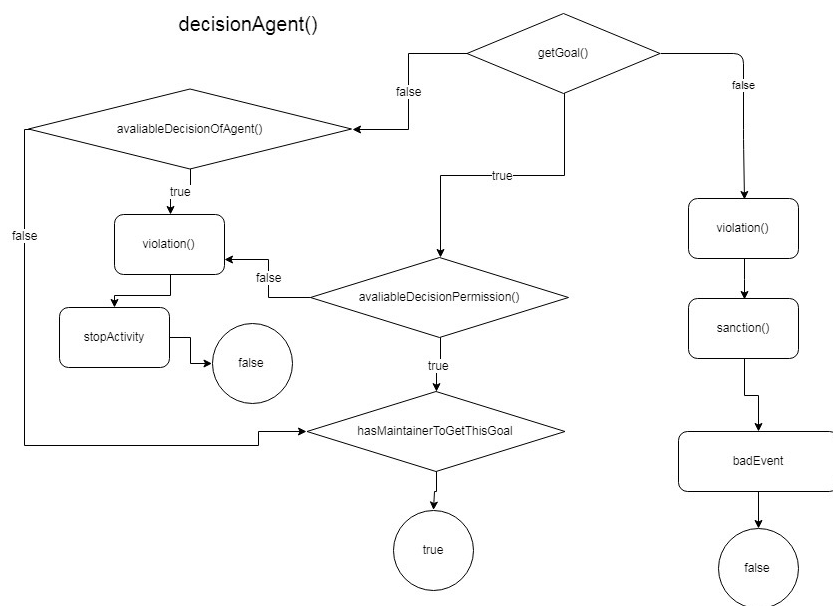
Definição 28

Para todos os agentes que receberam uma obrigação de executar g_{e_i} , ao cumprir com a condição $tryReach(g_{e_i}, ag) \wedge \neg stopOperation$ então $isCompleted(g_{e_i})$, se não, então é verdade $failed(g_{e_i})$

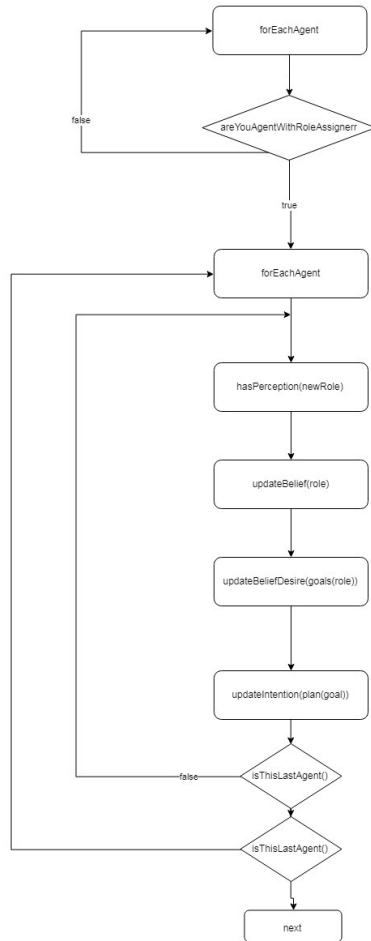
Algoritmo que implementa das Definições



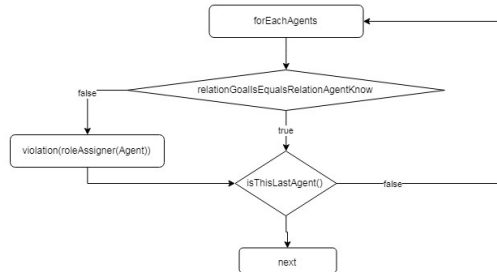




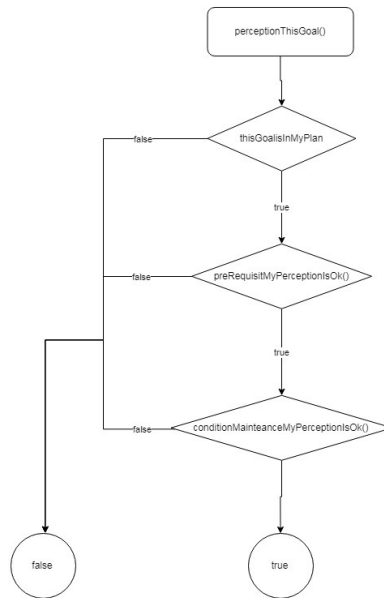
AttributionRoleToAgents



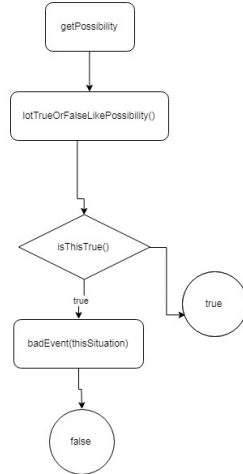
VerifyViolationAssigned



getGoal



randomicCondition()



AvailableDecisionOfAgent

