

Lógica Fuzzy

Lucas Tonussi¹, Thiago Senhorinha¹, Maurilio Atila
{lptonussi, thiago.senhorinha, cabelotaina}@gmail.com

¹Universidade Federal de Santa Catarina
Departamento de Ciência da Computação

1. Introdução

Neste trabalho desenvolvemos um controlador fuzzy de um motorista para estacionar um caminhão de ré numa doca, utilizando os conhecimentos sobre sistemas fuzzy adquiridos durante as aulas.

2. Sistema de Controle Fuzzy

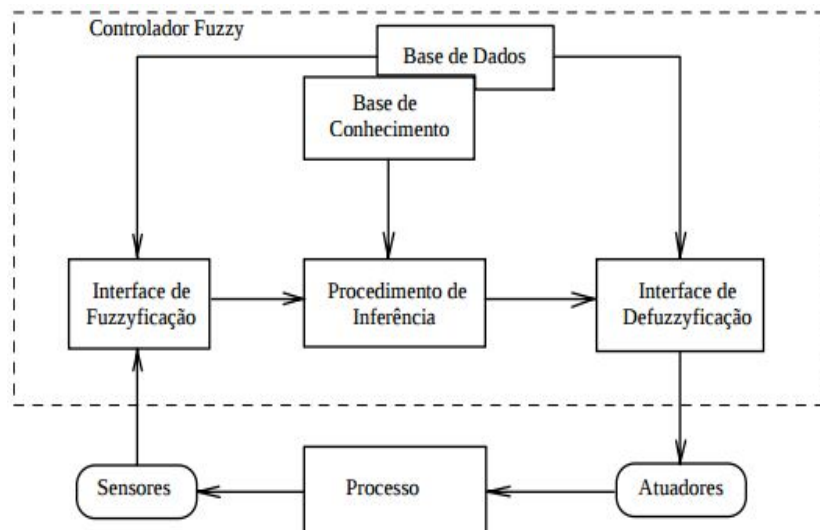
Resumidamente, a idéia em controle fuzzy é modelar e simular as ações físicas a partir de conhecimento de um especialista. A lógica fuzzy é uma lógica baseada na teoria dos conjuntos fuzzy, os conjuntos fuzzy dão a possibilidade de interpretar valores verdade que não tem correspondência explicitamente booleana, i.e. um interruptor de lâmpada pode ser verdadeiro ou falso. Já em lógica fuzzy uma variável pode compreender diferentes classes de verdade. de ser verdadeiro ou falso.

A lógica fuzzy (nebulosa), permite raciocínio aproximado sobre variáveis que representam situações reais tais como:

“controle de volante de um carro”. A princípio, uma máquina não sabe o quanto deve-se esterçar para esquerda, direita, ou permanecer em um meio termo, porém em lógica fuzzy é possível aplicar método de defuzzyficação para quantificar que retorna um valor numérico sensível às classes criadas para representar modos de raciocínio (i.e esterçar para

direita, esquerda, permanecer em um meio termo). A estrutura de um processo controlado por um controlador fuzzy é mostrada na figura ao lado

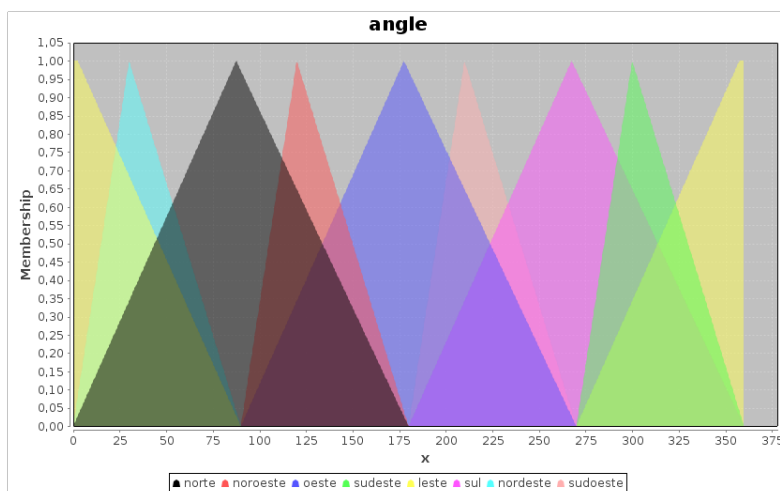
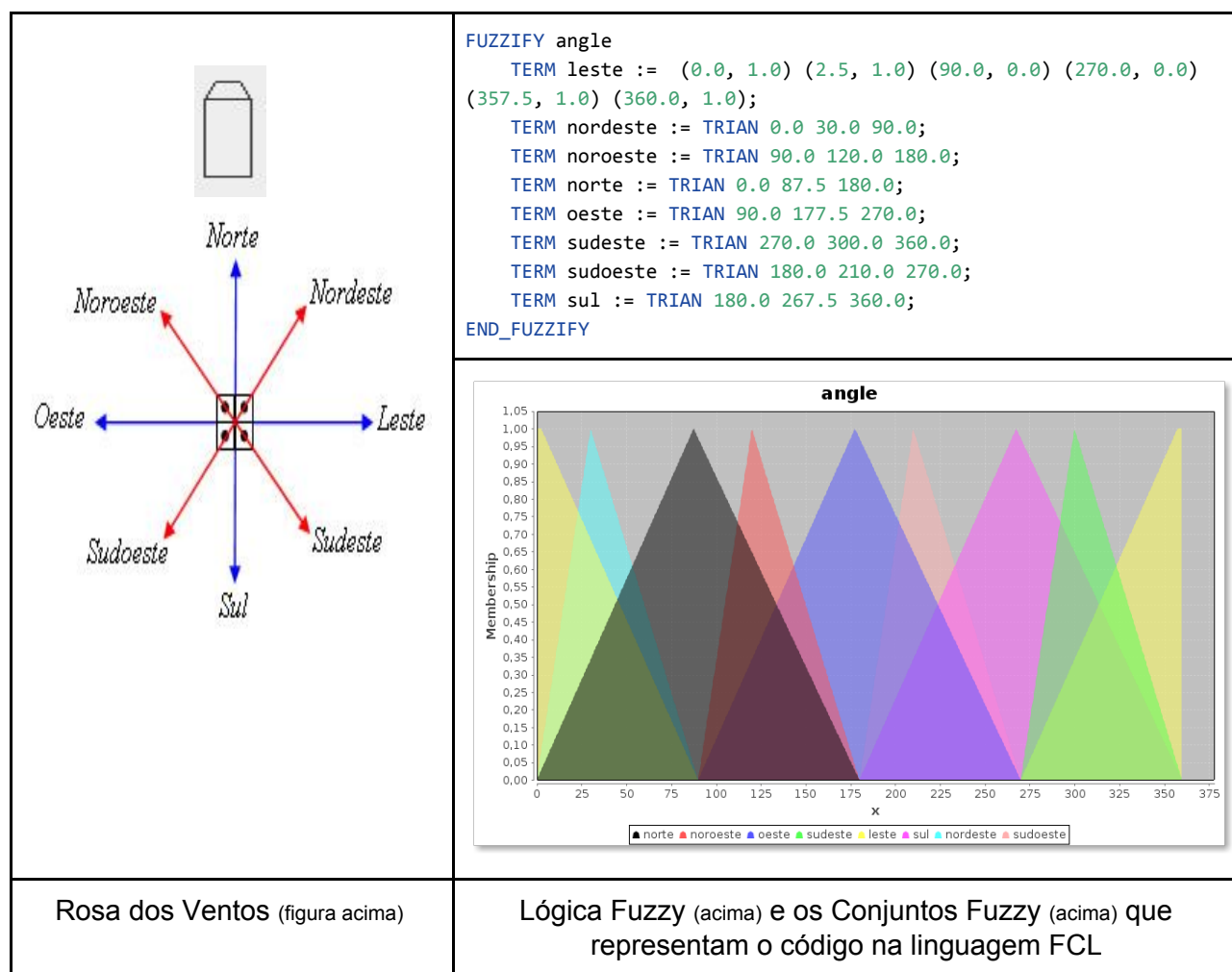
A relação dos conjuntos fuzzy e a regras é dada por: Regra i : Se x é A_i Então $f(x)$ é B_i , $i = 1, \dots, N$, onde x , representa uma variável independente. No capítulo 8, foi disponibilizado o código



em linguagem FCL, as regras, no bloco de regras, seguem essa formulação. No caso do sistema de estacionamento do caminhão: x , y , e $angle$ (ângulo) são as variáveis independentes do sistema.

3. Conjuntos

A fuzzyficação abaixo representa a localização angular do caminhão no plano 2D. Leste precisou ser feito ponto a ponto, pois o conjunto *leste* está fechando o comprimento de ângulos de 360° . Essas classes (ou conjuntos fuzzy) representam a rosa dos ventos (à esquerda). Com essas classes é possível verificar para onde a frente do caminhão está apontada.



O volante é representado pelas seguintes classes fuzzy (abaixo). A linguagem FCL (através da biblioteca JFuzzyLogic) oferece alguns métodos para quantificar valores de conjuntos fuzzy. Estamos aplicando o quantificador CDA para o controle do volante. A seguir no capítulo 4, explicaremos o método CDA mais detalhadamente.

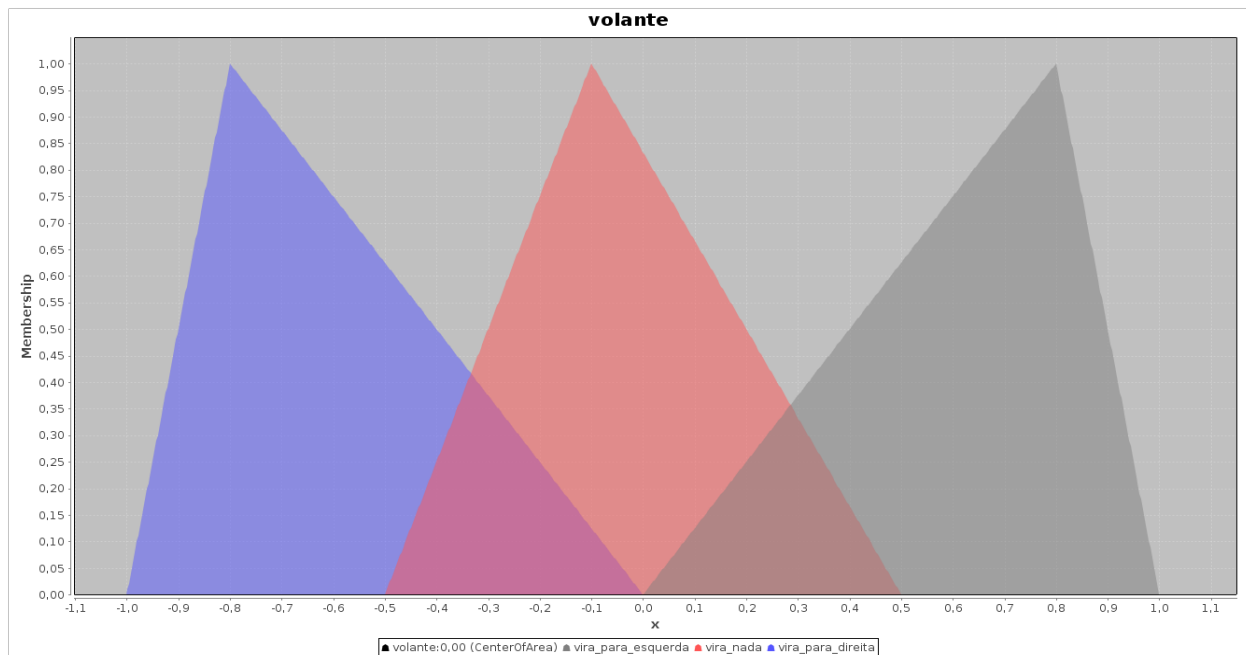


```

DEFUZZIFY volante
  TERM vira_nada := TRIAN -0.5 -0.1 0.5;
  TERM vira_para_direita := TRIAN -1.0 -0.8 0.0;
  TERM vira_para_esquerda := TRIAN 0.0 0.8 1.0;
  METHOD : COA;
  DEFAULT := 0.0;
  RANGE := (-1.0 .. 1.0);
END_DEFUZZIFY

```

Os extremos do intervalo (-1, 1) representam -30 graus e 30 graus para a rotação do volante.



Classes que representam a variável de manuseio do volante do caminhão.

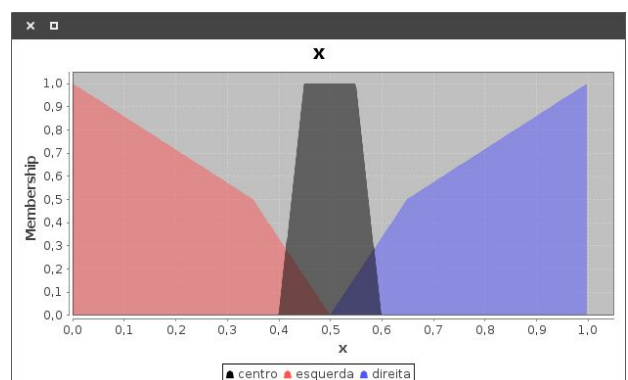
As variáveis independentes x e y estão descritas nas classes fuzzy, a seguir (figuras abaixo).

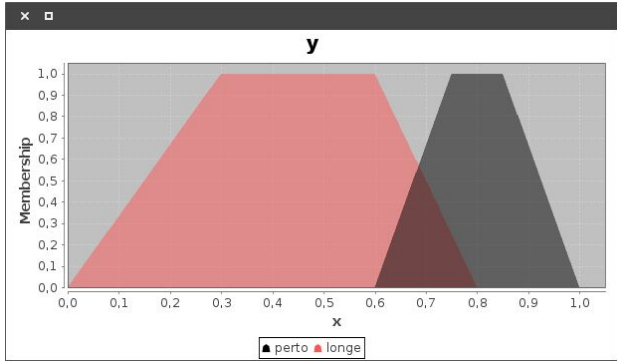
```

FUZZIFY x
  TERM centro := TRAPE 0.4 0.45 0.55 0.6;
  TERM direita := (0.5, 0.0) (0.65, 0.5) (1.0, 1.0);
  TERM esquerda := (0.0, 1.0) (0.35, 0.5) (0.5, 0.0);
END_FUZZIFY

```

Separamos a eixo horizontal em 3 intervalos. Como a baliza está fixada no centro, modelamos os conjuntos para que quando o veículo estivesse nos cantos, ele deve manobrar em direção ao centro.



	Conjuntos que representam o espaço da variável independente 'x'.
<pre> FUZZIFY y TERM longe := TRAPE 0.0 0.3 0.6 0.8; TERM perto := TRAPE 0.6 0.75 0.85 1.0; END_FUZZIFY </pre> <p>Esses conjuntos definem o quão perto estamos da baliza. Precisamos defini-los para conseguir o afastamento da ponto desejado quando o caminhão não houvesse espaço suficiente para manobra.</p>	 <p>Conjuntos que representam o espaço da variável independente 'y'.</p>

4. Método de defuzzificação e Regras

Utilizamos o método de defuzzificação chamado Centro de Área, o qual será explicado a seguir. Apesar de não haver nenhum procedimento sistemático para a escolha da estratégia de defuzzificação (visto em aula). “O método de Defuzzificação do Centro de Área procura calcular (usando a fórmula abaixo) o “melhor” harmonização possível entre vários conjuntos (ou classes) que compõem uma variável de saída (i.e. volante do caminhão).”¹.

$$Centro\ de\ Área = \frac{\sum_{k=1}^n \mu_{\beta}(v_k) \times v_k}{\sum_{k=1}^n \mu_{\beta}(v_k)}, \text{ onde,}$$

$$v_k^{\uparrow} \in v^{\uparrow} \subseteq v, v^{\uparrow} = \{v_k^{\uparrow} \mid \mu_{\beta}(v_k^{\uparrow}) = \max_j(\mu_{\beta}(v_j)), v_j \in V\}.^2$$

As funções de defuzzificação servem para retornar um valor numérico para as classes fuzzy. Em outras palavras: “A interface de defuzzificação transforma as ações de controle fuzzy inferidas em ações de controle não-fuzzy” [GOMIDE *et al.* 1994].

5. Dificuldades encontradas

Encontramos dificuldades em estacionar o caminhão quando ele estava muito próximo ou nas diagonais inferiores da baliza. Visto que não existe a possibilidade do caminhão andar para frente, optamos por deslocarmos o veículo para longe da baliza para obter espaço para manobrar quando ele estivesse dentro do conjunto “**perto**” (fuzzyficação de ‘y’), como mostram as figuras abaixo:

¹ Exemplo do método CDA sendo aplicado para o giro do volante. Disponível em: http://zone.ni.com/reference/en-XX/help/370401H-01/lvpidmain/center_of_area/.

² Seta para cima é a notação de Knuth para inteiros muito grandes, Disponível em: https://en.wikipedia.org/wiki/Knuth's_up-arrow_notation.

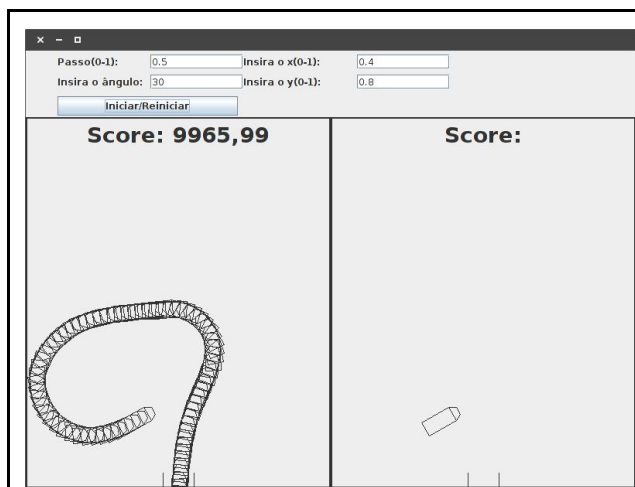


Figura mostrando um exemplo (1) das regras causando a convergência do caminhão.

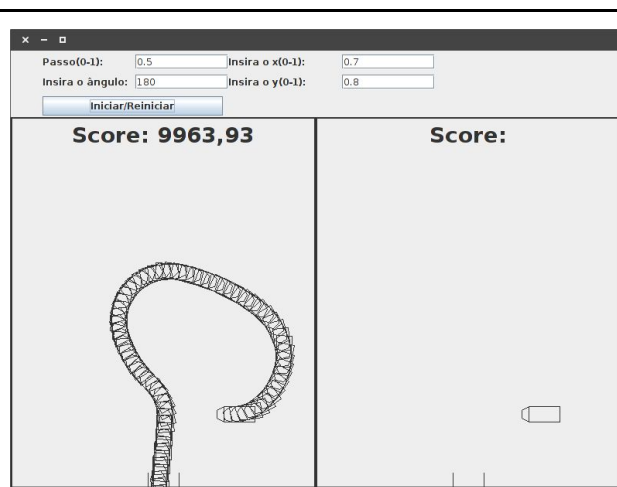


Figura mostrando um exemplo (2) das regras causando a convergência do caminhão.

6. Conclusões

Apesar de algumas dificuldades em ajustar as regras e os conjuntos da lógica difusa, conseguimos estacionar o veículo com êxito.

Percebemos que a lógica fuzzy dá aos computadores a capacidade de emular o raciocínio humano e resolver certos tipos de problemas de forma eficiente, porém ficou evidente algumas limitações em seu uso. Uma delas é a sua utilização em sistemas que precisam de extrema precisão, visto que a lógica trabalha com aproximações. Também percebemos que a lógica criada é extremamente dependente de um especialista e pelo fato de serem estáticos, sistemas fuzzy não conseguem “aprender”.

7. Referencias

GOMIDE, F. A. C.; GUDWIN, R. R.; MODELAGEM, CONTROLE, SISTEMAS E LÓGICA FUZZY Departamento de Engenharia de Computação e Automação Industrial (DCA) Faculdade de Engenharia Elétrica (FEE) Universidade Estadual de Campinas (UNICAMP) CP 6101, CEP 13081-970, Campinas - SP e-mail : gomide@dca.fee.unicamp.br gudwin@dca.fee.unicamp.br. SBA Controle & Automação, vol. 4, n. 3, Setembro, 1994.

8. Código FCL

```
FUNCTION_BLOCK driver

VAR_INPUT
    angle : REAL;
    x : REAL;
    y : REAL;
END_VAR

VAR_OUTPUT
    volante : REAL;
END_VAR

FUZZIFY angle
    TERM leste := (0.0, 1.0) (2.5, 1.0) (90.0, 0.0) (270.0, 0.0) (357.5, 1.0) (360.0, 1.0);
    TERM nordeste := TRIAN 0.0 30.0 90.0;
    TERM noroeste := TRIAN 90.0 120.0 180.0;
    TERM norte := TRIAN 0.0 87.5 180.0;
    TERM oeste := TRIAN 90.0 177.5 270.0;
    TERM sudeste := TRIAN 270.0 300.0 360.0;
    TERM sudoeste := TRIAN 180.0 210.0 270.0;
    TERM sul := TRIAN 180.0 267.5 360.0;
END_FUZZIFY

FUZZIFY x
    TERM centro := TRAPE 0.4 0.45 0.55 0.6;
    TERM direita := (0.5, 0.0) (0.65, 0.5) (1.0, 1.0) ;
    TERM esquerda := (0.0, 1.0) (0.35, 0.5) (0.5, 0.0) ;
```

END_FUZZIFY

FUZZIFY y

TERM longe := TRAPE 0.0 0.3 0.6 0.8;
TERM perto := TRAPE 0.6 0.75 0.85 1.0;

END_FUZZIFY

DEFUZZIFY volante

TERM vira_nada := TRIAN -0.5 -0.1 0.5;
TERM vira_para_direita := TRIAN -1.0 -0.8 0.0;
TERM vira_para_esquerda := TRIAN 0.0 0.8 1.0;
METHOD : COA;
DEFAULT := 0.0;
RANGE := (-1.0 .. 1.0);

END_DEFUZZIFY

RULEBLOCK No1

ACT : MIN;
ACCU : MAX;
AND : MIN;

RULE 1 : IF ((x IS esquerda) AND (y IS perto)) AND (angle IS leste) THEN volante IS
vira_para_direita;
RULE 2 : IF ((x IS esquerda) AND (y IS perto)) AND (angle IS norte) THEN volante IS
vira_para_esquerda;
RULE 3 : IF ((x IS esquerda) AND (y IS perto)) AND (angle IS nordeste) THEN volante IS
vira_para_direita;
RULE 4 : IF (x IS esquerda) AND (angle IS noroeste) THEN volante IS vira_nada;
RULE 5 : IF (x IS esquerda) AND (angle IS oeste) THEN volante IS vira_nada;
RULE 6 : IF (x IS esquerda) AND (angle IS sul) THEN volante IS vira_para_direita;
RULE 7 : IF (x IS esquerda) AND (angle IS sudeste) THEN volante IS vira_para_direita;
RULE 8 : IF (x IS esquerda) AND (angle IS sudoeste) THEN volante IS vira_para_direita;
RULE 9 : IF ((x IS esquerda) AND (y IS longe)) AND (angle IS leste) THEN volante IS
vira_para_esquerda;
RULE 10 : IF ((x IS esquerda) AND (y IS longe)) AND (angle IS norte) THEN volante IS
vira_para_esquerda;
RULE 11 : IF ((x IS esquerda) AND (y IS longe)) AND (angle IS nordeste) THEN volante IS
vira_para_esquerda;
RULE 12 : IF (x IS centro) AND (angle IS leste) THEN volante IS vira_para_esquerda;
RULE 13 : IF (x IS centro) AND (angle IS oeste) THEN volante IS vira_para_direita;
RULE 14 : IF (x IS centro) AND (angle IS nordeste) THEN volante IS vira_para_esquerda;
RULE 15 : IF (x IS centro) AND (angle IS noroeste) THEN volante IS vira_para_direita;

RULE 16 : IF ((x IS direita) AND (y IS perto)) AND (angle IS oeste) THEN volante IS
vira_para_esquerda;
RULE 17 : IF ((x IS direita) AND (y IS perto)) AND (angle IS norte) THEN volante IS
vira_para_direita;
RULE 18 : IF ((x IS direita) AND (y IS perto)) AND (angle IS noroeste) THEN volante IS
vira_para_esquerda;
RULE 19 : IF (x IS direita) AND (angle IS nordeste) THEN volante IS vira_nada;
RULE 20 : IF (x IS direita) AND (angle IS leste) THEN volante IS vira_nada;
RULE 21 : IF (x IS direita) AND (angle IS sul) THEN volante IS vira_para_esquerda;
RULE 22 : IF (x IS direita) AND (angle IS sudeste) THEN volante IS vira_para_esquerda;
RULE 23 : IF (x IS direita) AND (angle IS sudoeste) THEN volante IS vira_para_esquerda;
RULE 24 : IF ((x IS direita) AND (y IS longe)) AND (angle IS oeste) THEN volante IS
vira_para_direita;
RULE 25 : IF ((x IS direita) AND (y IS longe)) AND (angle IS norte) THEN volante IS
vira_para_direita;
RULE 26 : IF ((x IS direita) AND (y IS longe)) AND (angle IS noroeste) THEN volante IS
vira_para_direita;

RULE 27 : IF x IS centro OR x IS direita AND angle IS sudeste THEN volante IS vira_para_esquerda;
RULE 28 : IF x IS centro OR x IS direita AND angle IS sudoeste THEN volante IS vira_para_esquerda;

```
RULE 29 : IF x IS centro OR x IS esquerda AND angle IS sudoeste THEN volante IS vira_para_direita;  
RULE 30 : IF x IS centro OR x IS esquerda AND angle IS sudeste THEN volante IS vira_para_direita;
```

```
END_RULEBLOCK
```

```
END_FUNCTION_BLOCK
```