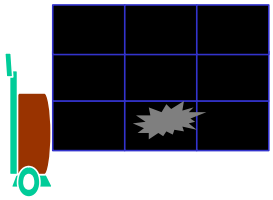
Tema 2 (II): Representación del Conocimiento. Representación Basada en Reglas

Ejemplo De Sistemas Basados En Reglas (I): Aspiradora

Sistemas de Representación y Procesamiento Automático del Conocimiento



Control del movimiento de una aspiradora por una habitación vacía para limpiar la suciedad

- **Ontología**:
  - (dirt <punto>): La baldosa <punto> está sucia.
    - Puntos representados por p*xy*. Ejm.: p01
  - (in <punto>): La aspiradora se encuentra en la baldosa <punto>
  - (facing <orientación>): La aspiradora está orientada hacia <orientación>.
    - Orientación toma valores en (n, s, e, w)

Baldosas Sucias

(dirt p02)
(dirt p12)

Aspiradora

(in p00)
(facing n)

		2, 2
0, 1	1, 1	2, 1
0, 0	1, 0	2, 0

Hechos

Limpiar Baldosa

```
(defrule limpieza
  (in ?xy)
  ?f <- (dirt ?xy)
  =>
  (printout t "Limpie un cuadro..." crlf)
  (retract ?f))
```

0, 2	1, 2	2, 2
0, 1	1, 1	2, 1
0, 0	1, 0	2, 0

Reglas

Movimiento hacia el Norte

```
(defrule navega00n
  ?f <- (in p00)
  (not (dirt p00))
  (facing n)
  =>
  (assert (in p01))
  (retract ?f) )
```

0, 2	1, 2	2, 2
0, 1	1, 1	2, 1
0, 0	1, 0	2, 0

```
(defrule navega01n
  ?f <- (in p01)
  (facing n)
  (not (dirt p01))
  =>
  (assert (in p02))
  (retract ?f) )
```

0, 2	1, 2	2, 2
0, 1	1, 1	2, 1
0, 0	1, 0	2, 0

Reglas

Cambio Orientación

```
(defrule navega02n
  (in p02)
  ?f <- (facing n)
  (not (dirt p02))
  =>
  (assert (facing e))
  (retract ?f) )
```

0, 2	1, 2	2, 2
0, 1	1, 1	2, 1
0, 0	1, 0	2, 0

Reglas

```
(defrule navega02e
  ?f <- (in p02)
  (facing e)
  (not (dirt p02))
  =>
  (assert (in p12))
  (retract ?f) )

(defrule navega12e
  (in p12)
  ?f <- (facing e)
  (not (dirt p12))
  =>
  (assert (facing s))
  (retract ?f) )

(defrule navega12s
  ?f <- (in p12)
  (facing s)
  (not (dirt p12))
  =>
  (assert (in p11))
  (retract ?f) )

(defrule navega11s
  ?f <- (in p11)
  (facing s)
  (not (dirt p11))
  =>
  (assert (in p10))
  (retract ?f) )

(defrule navega10s
  (in p10)
  ?f <- (facing s)
  (not (dirt p10))
  =>
  (assert (facing e))
  (retract ?f) )

(defrule navega10e
  ?f <- (in p10)
  (facing e)
  (not (dirt p10))
  =>
  (assert (in p20))
  (retract ?f) )

(defrule navega20e
  (in p20)
  ?f <- (facing e)
  (not (dirt p00))
  =>
  (assert (facing n))
  (retract ?f) )

(defrule navega20n
  ?f <- (in p20)
  (facing n)
  (not (dirt p20))
  =>
  (assert (in p21))
  (retract ?f) )
```

Reglas

```
(defrule navega21n
  ?f <- (in p21)
  (facing n)
  (not (dirt p21))
  =>
  (assert (in p22))
  (retract ?f) )

(defrule navega22w
  ?f <- (in p22)
  (facing w)
  (not (dirt p22))
  =>
  (assert (in p12))
  (retract ?f) )

(defrule navega02w
  (in p02)
  ?f <- (facing w)
  (not (dirt p02))
  =>
  (assert (facing s))
  (retract ?f) )

(defrule navega01s
  ?f <- (in p01)
  (facing s)
  (not (dirt p01))
  =>
  (assert (in p00))
  (retract ?f) )

(defrule navega22n
  (in p22)
  ?f <- (facing n)
  (not (dirt p22))
  =>
  (assert (facing w))
  (retract ?f) )

(defrule navega12w
  ?f <- (in p12)
  (facing w)
  (not (dirt p12))
  =>
  (assert (in p02))
  (retract ?f) )

(defrule navega02s
  ?f <- (in p02)
  (facing s)
  (not (dirt p02))
  =>
  (assert (in p01))
  (retract ?f) )
```

Reglas

- **Ontología**:

(MAX_X <valor>): Las coordenadas x de la habitación están en el rango [0, <valor>]

(MAX_Y <valor>): Las coordenadas y de la habitación están en el rango [0, <valor>]

(punto x <valor_x> y <valor_y> estado sucio): La baldosa en las coordenadas [<valor_x>, <valor_y>] está sucia

(aspiradora x <valor_x> y <valor_y> direccion <orientacion>):
La aspiradora se encuentra en las coordenadas [<valor_x>,<valor_y>] con orientación <orientacion>

Tamaño habitación

(MAX_X 2)
(MAX_Y 2)

Localización Suciedad

(punto x 0 y 2 estado sucio)
(punto x 1 y 1 estado sucio)

Localización Aspiradora

(aspiradora x 0 y 0 direccion norte)

Hechos

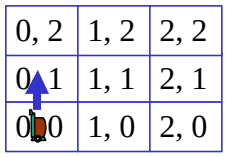
Limpiar

```
(defrule limpieza
  (aspiradora x ?x y ?y $?)
  ?f <- (punto x ?x y ?y estado sucio)
  =>
  (printout t "Limpie la baldosa " ?x ", " ?y "..." crlf)
  (retract ?f))
```

Reglas

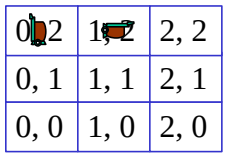
Movimiento hacia el Norte

```
(defrule navegaN
  ?f <- (aspiradora x ?x y ?y direccion norte)
  (not (punto x ?x y ?y estado sucio))
  (MAX_Y ?max)
  (test (< ?y ?max))
  =>
  (retract ?f)
  (assert (aspiradora x ?x y (+ ?y 1) direccion norte))
  (printout t "Nueva posicion: " ?x ", " (+ ?y 1) "..." crlf))
```



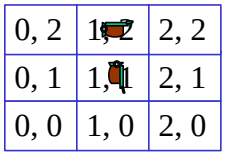

```
(defrule navegaN_fin
  ?f <- (aspiradora x ?x y ?y direccion norte)
  (not (punto x ?x y ?y estado sucio))
  (MAX_Y ?yMax)
  (test (= ?y ?yMax))
  (MAX_X ?xMax)
  (test (< ?x ?xMax))
  =>
  (retract ?f)
  (assert (aspiradora x (+ ?x 1) y ?y direccion este))
  (printout t "Nueva posicion: " (+ ?x 1) ", " ?y "..." crlf))
```

Cambio Orientación



Reglas

Cambio Orientación

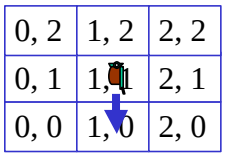



```
(defrule navegaE_alS
  ?f <- (aspiradora x ?x y ?y direccion este)
  (not (punto x ?x y ?y estado sucio))
  (test (> ?y 0))
  =>
  (retract ?f)
  (assert (aspiradora x ?x y (- ?y 1) direccion sur))
  (printout t "Nueva posicion: " ?x ", " (- ?y 1) "..." crlf))
```

Reglas

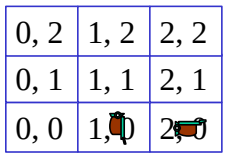
Movimiento hacia el Sur

```
(defrule navegaS
  ?f <- (aspiradora x ?x y ?y direccion sur)
  (not (punto x ?x y ?y estado sucio))
  (test (> ?y 0))
  =>
  (retract ?f)
  (assert (aspiradora x ?x y (- ?y 1) direccion sur))
  (printout t "Nueva posicion: " ?x ", " (- ?y 1) "..." crlf))
```



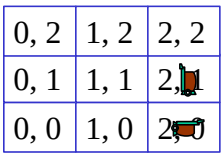

```
(defrule navegaS_fin
  ?f <- (aspiradora x ?x y 0 direccion sur)
  (not (punto x ?x y 0 estado sucio))
  (MAX_X ?xMax)
  (test (< ?x ?xMax))
  =>
  (retract ?f)
  (assert (aspiradora x (+ ?x 1) y 0 direccion este))
  (printout t "Nueva posicion: " (+ ?x 1) ", 0..." crlf))
```

Cambio Orientación



Reglas

Cambio Orientación




```
(defrule navegaE_alN
  ?f <- (aspiradora x ?x y ?y direccion este)
  (not (punto x ?x y ?y estado sucio))
  (test (= ?y 0))
  =>
  (retract ?f)
  (assert (aspiradora x ?x y (+ ?y 1) direccion norte))
  (printout t "Nueva posicion: " ?x ", " (+ ?y 1) "..." crlf))
```

Reglas