Desarrollo de un Sistema Experto:

Personas mayores que viven en solas

Juan Manuel Castillo Nievas

Resumen

Este Sistema Experto está dedicado para alertar de los problemas que pueden tener las personas mayores que viven solas. Se trata de crear un Sistema Basado en el Conocimiento para una persona que vive sola en casa, salvo una visita de una asistente de lunes a sábado de 10:00 a 14:00. La casa contiene sensores de movimiento en todas las habitaciones y un sensor magnético para las puertas del cuarto de baño, cocina, calle, salón y dormitorio.

Se dispone de un simulador en CLIPS para simular las acciones que puede realizar una persona mayor en la casa. Este simulador es igual al utilizado en la práctica 1 pero con una complejidad menor, pues las funcionalidades ahora son distintas.

El fichero principal es el llamado *CasaInteligente.clp*, que contiene todos los hechos y reglas necesarios para que el Sistema Experto funcione adecuadamente. Este fichero está comentado correctamente y las reglas y hechos siguen una estructura que se explicará en los apartados siguientes. El manual de uso de este simulador se encuentre en el último apartado del documento.

Descripción del proceso seguido

Procedimiento seguido para el desarrollo de la base de conocimiento

Para la base del conocimiento se ha reutilizado parte del conocimiento que se usó para la práctica 1. En concreto, se han reutilizado los siguientes conceptos extraídos del conocimiento del profesor:

- Una habitación está activa cuando el sensor de movimiento de la habitación está en ON
- Cuando un sensor de movimiento indica OFF en una habitación, esa habitación pasa al estado parece inactiva
- Cuando una habitación parece inactiva durante más de 10 segundos, esa habitación pasa a estar inactiva
- Cuando se dispara el sensor de movimiento de una habitación, se registra que se ha podido producir un paso desde las habitaciones que sean accesibles y que estuvieran activas o recientemente activas
- Cuando sólo hay un posible paso a una habitación, se deduce que se ha producido el paso

El conocimiento sobre las 10 funcionalidades que se han implementado es un conocimiento propio, extraído a partir de mis propios conocimientos utilizando el sentido común dada cada situación. Cada funcionalidad será explicada con detalle más adelante, pero por poner un ejemplo: una persona mayor tiene menos actividad de lo normal cuando realiza menos pasos entre habitaciones que los que hace habitualmente a una determinada hora.

Procedimiento de validación y verificación del sistema

Para el proceso de validación se ha interactuado con el sistema con diferentes situaciones para comprobar si, efectivamente, la respuesta era coherente con la situación. El simulador cuenta con un fichero *DatosSimulados.txt* que contiene la activación/desactivación de los sensores de una habitación a una determinada hora, tal y como se hacía en la práctica 1. Se han probado las situaciones que normalmente se dan en una persona mayor, no se han probado situaciones que sean imposibles de realizar. También es fácil validar el sistema ya que está dividido en módulos y no puede haber interacciones entre los módulos.

Para verificar que el sistema daba una respuesta coherente en cada situación, he dejado que varias personas de mi entorno modifiquen el fichero *DatosSimulados.txt* para que pongan diferentes situaciones para ver si el sistema daba una respuesta correcta.

Descripción del sistema

Variables de entrada del problema

Cuando se lanza el simulador, se pregunta al usuario el número de segundos que se quieren simular a partir de una determinada hora. La hora de inicio del simulador se puede modificar en el fichero *SituacionInicial.txt*. Se indica a través del hecho *(simulado desde-las hh mm ss segundosporciclo 1)*. Si la hora de inicio es las 10:00:00 y el usuario indica al simulador que se simulen 3600 segundos, el simulador funcionará desde las 10:00:00 hasta las 11:00:00.

Variables de salida del problema

Como salida del sistema se mostrarán comentarios cuando se produzcan ciertas anomalías de acuerdo a las funcionalidades. Por ejemplo, si la persona mayor ha ido varias veces al baño en las últimas 3 horas, el sistema mostrará un comentario informando al usuario de dicha situación. También se muestran comentarios por pantalla cada vez que se activa/desactiva el sensor de movimiento o magnético de cada habitación o puerta. Cada comentario muestra la hora en formato hh:mm:ss en la que se lanza dicho comentario.

Conocimiento global del sistema

Inicialmente se van a cargar las variables globales (manejan la incertidumbre en cierto modo) y el conocimiento sobre la casa. Las variables globales que se cargan son:

- **hora_inicio_dia**: es la hora en la que empieza el día (en segundos). Para mí, el día empezaría a las 12:00 (43200 segundos).
- **hora_fin_dia**: es la hora en la que finaliza el día. Para mí, el día terminaría a las 20:00.
- **hora_inicio_maniana**: hora en la que empieza la franja de la mañana.
- **hora_fin_maniana**: hora en la que termina la franja de la mañana.
- **hora_inicio_tarde**: hora en la que empieza la franja de la tarde.
- **hora_fin_tarde**: hora en la que termina la franja de la tarde.
- **hora_inicio_noche**: hora en la que empieza la franja de la noche.
- **hora_fin_noche**: hora en la que termina la franja de la noche.
- **hora_inicio_noche_persona_despierta_noche**: una funcionalidad trata de saber si la persona mayor se ha levantado durante más de 15 minutos por la noche. La hora a partir de la cual se empieza a contar los minutos que la persona está despierta se especifica en esta variable.
- **hora_fin_noche_persona_despierta_noche**: la hora final de la noche en relación a la variable anterior.
- **hora_inicio_llegada_asistenta**: hora a partir de la cual la asistenta puede llegar a la casa.
- **hora_fin_llegada_asistenta**: hora límite para que la asistenta llegue a la casa. Si a esa hora la asistenta no ha llegado, se considera que la asistenta llega tarde.
- **hora_inicio_salida_asistenta**: hora a partir de la cual la asistenta puede irse de la casa.
- **hora_fin_salida_asistenta**: hora límite para que la asistenta se vaya de la casa.
- **hora_inicio_persona_durmiento_tarde**: hora a partir de la cual se considera que, si la persona no se ha ido a dormir, significa que se va a ir a dormir tarde.
- **hora_fin_persona_durmiento_tarde**: considero que a partir de esta hora es muy poco probable que la persona siga durmiendo. Como muy tarde, la persona se irá a dormir a esta hora.

- **n_veces_banio**: en esta variable se almacena el número de veces que la persona ha ido al baño en las últimas 3 horas. Se actualiza en cada segundo que transcurre en el simulador.
- **n_pasos**: en esta variable se cuenta el número de veces que la persona ha pasado de una habitación a otra. Se actualiza en cada segundo que transcurre en el simulador.
- **n_pasos_franja_maniana**: misma utilidad que *n_pasos* pero contando solo los pasos que se dan en la franja de por la mañana.
- **n_pasos_franja_tarde**: misma utilidad que *n_pasos* pero contando solo los pasos que se dan en la franja de por la tarde.
- **n_pasos_franja_noche**: misma utilidad que *n_pasos* pero contando solo los pasos que se dan en la franja de por la noche.

Los hechos que se cargan inicialmente en el sistema son los que representan el conocimiento sobre la casa. Van a aparecer los siguientes hechos:

- **Template (Habitacion (habitacion ?h) (ventanas ?n))**: representa una habitación ?h con un número de ventanas ?n
- **Template (Puerta (puerta_habitacion1 ?h1) (puerta_habitacion ?h2))**: representa una puerta que conecta la habitación ?h1 con la habitación ?h2.
- **Template (Paso (paso_habitacion1 ?h1) (paso_habitacion ?h2))**: representa una conexión de la habitación ?h1 con la habitación ?h2 pero sin una puerta.
- **Template (estado_habitacion (habitacion ?h) (estado ?a) (instante ?i))**: representa el estado de una habitación ?h en el instante ?i. Inicialmente, todas las habitaciones estarán en el estado *inactiva*.
- **(ultima_desactivacion movimiento ?h 0)**: cada habitación tendrá asociada una última desactivación. Inicialmente, todas las habitaciones tendrán la última desactivación en el instante 0.
- **(ultima_activación movimiento ?h 0)**: similar a lo anterior, pero con la última activación.
- **(numero_de_personas 1)**: el número de personas que hay en la casa. Inicialmente, hay 1 persona (la persona mayor). Este hecho se modificará cada vez que entre o salga una persona.
- **(dia_de_la_semana lunes)**: el día de la semana. Inicialmente, el día en el que empieza la simulación es el lunes.
- **(asistenta_en_casa false)**: si el valor del segundo argumento es *false*, significa que la asistenta no está en casa. Cuando la asistenta esté en casa, este hecho cambiará a (asistenta_en_casa true).
- **(NumeroPasosNormales ?n ?i)**: este hecho significa que desde las 00:00 hasta ?i, la persona mayor hace normalmente ?n pasos de una habitación a otra.

- **(NumeroPasosNormalesFranja ?n franja_maniana)**: significa que la persona mayor hace normalmente ?n pasos de una habitación a otra en la franja de por la mañana.
- **(NumeroPasosNormalesFranja ?n franja_tarde)**: significa que la persona mayor hace normalmente ?n pasos de una habitación a otra en la franja de por la tarde.
- **(NumeroPasosNormalesFranja ?n franja_noche)**: significa que la persona mayor hace normalmente ?n pasos de una habitación a otra en la franja de por la noche.

Los hechos importantes que van a ir apareciendo en el sistema y que se deben tener en cuenta son:

- **(valor_registrado ?instante movimiento ?habitacion ?estado)**: en el instante ?instante se ha activado/desactivado el sensor de movimiento de la habitación ?habitacion.
- **(valor_registrado ?instante luminosidad ?habitacion ?l)**: igual que el anterior, pero con el sensor de luminosidad (el sensor de movimiento detecta la luminosidad también).
- **(ultimo_registro ?m ?h ?i)**: el último registro del sensor ?m en la habitación ?h en el instante ?i.
- **(ultimo_registro ?m ?h1 ?h2 ?i)**: el último registro del sensor magnético de la puerta que conecta ?h1 con ?h2 en el instante ?i.

Los hechos que aparezcan debido a los módulos se explicarán detalladamente en cada uno de ellos.

Especificación de los módulos

Toda la implementación de los módulos está en el mismo archivo *CasaInteligente.clp*, pero está todo comentado correctamente y se diferencia claramente cada módulo.

Módulo 1: La persona ha salido de la casa

Este módulo se activa en cada iteración (cada segundo) del simulador. Las reglas que forman este módulo, ordenadas por orden en el que se van deduciendo las cosas, son las siguientes:

Regla 1

```
(defrule Funcionalidad1
        ?f <- (modulo persona_ha_salido_de_casa)
        (HoraActualizada ?i)
        (ultimo_registro puerta ?h1 ?h2 ?i)
        (valor_registrado ?i puerta ?h1 ?h2 off)
        (test (or (eq ?h1 calle) (eq ?h2 calle)))
        =>
        (assert (PuertaCalleAbierta ?i))
        (retract ?f)
)
```

Esta regla es la primera que se activa, pues recibe el nombre del módulo. Con esta regla se comprueba si el último registro del sensor magnético de una puerta es de la puerta de la calle. Si es así, se inserta en la base de hechos (PuertaCalleAbierta ?i), que significa que la puerta de la calle se ha abierto en el instante ?i.

Regla 2

```
(defrule DeducirEntradaACasa
        ?f <- (PuertaCalleAbierta ?i)
        (posible_pasar ?h calle)
        (estado_habitacion (habitacion ?h) (estado inactiva))
        =>
        (bind ?instante (totalsegundos ?*hora* ?*minutos* ?*segundos*))

        (assert (parece_que_alguien_entra ?h ?instante))
        (retract ?f)
)
```

Esta regla se activa cuando está el hecho (PuertaCalleAbierta ?i). Si la puerta de la calle está abierta y la habitación de la casa que conecta con la calle está inactiva, significa que alguien está

entrando desde la calle, pero no es seguro. Por eso, se inserta el hecho (parece_que_alguien_entra ?h ?instante), que significa que parece que alguien entra a la habitación ?h en el instante ?instante.

Regla 3

```
(defrule DeducirEntradaACasa2
        (HoraActualizada ?i)
        ?f <- (parece_que_alguien_entra ?h ?instante)
        ?n <- (numero_de_personas ?numero)
        (estado_habitacion (habitacion ?h) (estado activa) (instante ?i))
        (test (< (- ?i ?instante) 3))
        (test (> (- ?i ?instante) 0))
        (dia_de_la_semana ?dia)
        ?f2 <- (asistenta_en_casa ?v)

        (hora_actual ?h1)
        (minutos_actual ?m1)
        (segundos_actual ?s1)
        =>

        (if (and (eq ?v false) (eq ?numero 1) (> ?i ?*hora_inicio_llegada_asistenta*) (< ?i
?*hora_fin_llegada_asistenta*) (not (eq ?dia domingo))) then
                (printout t ?h1 ":" ?m1 ":" ?s1 ": Ha entrado la asistenta" crlf)
                (retract ?f2)
                (assert (asistenta_en_casa true))

        else
                (if (eq ?numero 0) then
                (printout t ?h1 ":" ?m1 ":" ?s1 ": Ha entrado la persona mayor" crlf)

                else

                (printout t ?h1 ":" ?m1 ":" ?s1 ": Ha entrado otra persona" crlf)
                )
        )

        (retract ?f)
        (retract ?n)
        (assert (numero_de_personas (+ ?numero 1)))
)
```

Esta regla se activa cuando parece que alguien está entrando en la casa. Si la habitación que conecta con la calle se activa durante los próximos 3 segundos (y la habitación no estaba activa antes, cosa que se dedujo en la regla anterior), significa que ha entrado una persona:

- Si la asistenta no está en casa, no es domingo, solo hay 1 persona en casa, y la hora actual está en el intervalo de la hora de llegada de la asistenta, deduzco que la asistenta ha entrado.
- En otro caso, si no hay ninguna persona en casa, deduzco que es la persona mayor quien entra.
- En otro caso, deduzco que es otra persona que no es ni la persona mayor ni la asistenta.

En cualquier caso, el hecho (numero_de_personas ?n) se incrementa en 1.

Regla 4

```
(defrule DeducirEntradaACasa3
        ?f <- (parece_que_alguien_entra ?h ?instante)
        (HoraActualizada ?i)
        (test (> (- ?i ?instante) 3))
        =>
        (retract ?f)
)
```

Esta regla se utiliza para eliminar los hechos (parece_que_alguien_entra) de hace más de 3 segundos. Si han pasado más de 3 segundos y no he deducido que alguien ha entrado, elimino la posibilidad de que alguien vaya a entrar.

Regla 5

```
(defrule DeducirSalidaACalle
        ?f <- (PuertaCalleAbierta ?instante)
        (posible_pasar ?h calle)
        (estado_habitacion (habitacion ?h) (estado activa) (instante ?i))
        (test (< ?i ?instante))
        ?n <- (numero_de_personas ?numero)
        (dia_de_la_semana ?dia)
        ?f2 <- (asistenta_en_casa ?v)

        (hora_actual ?h1)
        (minutos_actual ?m1)
        (segundos_actual ?s1)
        =>

        (if (and (eq ?v true) (> ?i ?*hora_inicio_salida_asistenta*) (< ?i
?*hora_fin_salida_asistenta*) (not (eq ?dia domingo))) then
                (printout t ?h1 ":" ?m1 ":" ?s1 ": Ha salido la asistenta" crlf)
                (retract ?f2)
                (assert (asistenta_en_casa false))

        else
                (if (eq ?numero 1) then
                (printout t ?h1 ":" ?m1 ":" ?s1 ": La persona mayor ha salido a la calle" crlf)

                else
                (printout t ?h1 ":" ?m1 ":" ?s1 ": Ha salido otra persona" crlf)
                )
        )
        (retract ?f)
        (retract ?n)
        (assert (numero_de_personas (- ?numero 1)))
```

```
)
```

Esta regla se utiliza para deducir la salida a la calle. Si la puerta de la calle está abierta y la habitación que conecta con la calle está activa, deduzco que alguien va a salir a la calle:

- Si la asistenta está en casa, el número de personas en casa es 2, no es domingo, y la hora actual está en el intervalo de la hora de salida de la asistenta, deduzco que es la asistenta quien sale de la casa.
- En otro caso, si el número de personas en casa es 1, deduzco que es la persona mayor quien sale de casa.
- En otro caso, deduzco que es otra persona quien sale a la casa.

  En cualquier caso, el hecho (numero_de_personas ?n) se decrementa en 1.

Validación

Para validar, algunos de los escenarios probados han sido:

Esto simularía la entrada de la asistenta siempre y cuando no sea domingo:

```
(datosensor 10 00 0 puerta entrada calle off)
(datosensor 10 00 3 movimiento entrada on)
(datosensor 10 01 3 movimiento entrada off)
```

Esto simularía la salida de la asistenta siempre y cuando no sea domingo:

```
(datosensor 14 00 0 movimiento entrada on)
(datosensor 14 00 3 puerta entrada calle off)
(datosensor 14 01 0 movimiento entrada off)
```

Módulo 2: Siendo de día, la persona no se ha movido por la casa en las 3 últimas horas

Este módulo se activa en cada iteración (cada segundo) del simulador. La hora que abarca la frase *siendo de día* es la hora especificada en las variables globales *hora_inicio_dia* y *hora_fin_dia*. Para este módulo se utiliza un nuevo hecho llamado (ultimo_movimiento ?instante), que mantiene en la base de hechos el último movimiento captado por el sensor de cualquier habitación de la persona mayor. Esto quiere decir que la persona mayor debe estar sola en casa para que se registre el último movimiento y además la hora del sistema debe estar entre *hora_inicio_dia* y *hora_fin_dia*. Este módulo está compuesto por dos reglas únicamente:

Regla 1

```
(defrule Funcionalidad2
        ?f <- (modulo persona_no_se_ha_movido)
        (HoraActualizada ?horaactual)
        ?f2 <- (ultimo_movimiento ?i)
        (test (> (- ?horaactual ?i) 10800))
        (numero_de_personas ?numero)

        (hora_actual ?h1)
        (minutos_actual ?m1)
        (segundos_actual ?s1)
        =>

        (if (and (> ?horaactual ?*hora_inicio_dia*) (< ?horaactual ?*hora_fin_dia*) (eq
?numero 1))
        then
                (printout t ?h1 ":" ?m1 ":" ?s1 ": La persona lleva sin moverse más de 3
horas" crlf)
        )

        (retract ?f)
        (retract ?f2)
)
```

Esta regla se activa en cada segundo que pasa, pues ya he comentado anteriormente que el módulo se activa en cada iteración. Esta regla comprueba si el último movimiento de la persona mayor se realizó hace más de 3 horas (10800 segundos). Además, solamente se imprime en pantalla el aviso si es de día.

Regla 2

```
(defrule BorrarFuncionalidad2
        ?f <- (ultimo_movimiento ?i)
        (HoraActualizada ?horaactual)

        (test (and (< ?horaactual **hora_inicio_dia*) (> ?horaactual **hora_fin_dia*)))

        =>

        (retract ?f)
)
```

Esta regla se encarga de eliminar el último movimiento registrado cuando no es de día, pues ya no hará falta y habrá que esperar a que sea de día de nuevo para registrar el último movimiento.

Validación

Esto simularía que la persona no se ha movido en 3 horas, siempre y cuando esté sola en casa:

```
(datosensor 16 00 0 movimiento salon on)
```

A las 19:00, aparecía un mensaje en el sistema alertando de que la persona no se ha movido.

Módulo 3: La persona se ha despertado durante más de 15 minutos por la noche

Este módulo se activa cuando se registra un movimiento de la persona (estando sola en casa) cuando la hora actual del sistema está entre *hora_inicio_noche_persona_despierta_noche* y *hora_fin_noche_persona_despierta_noche*. Para este módulo se utiliza un nuevo hecho llamado (primer_movimiento_noche ?instante), que mantiene en la base de hechos el primer movimiento captado por el sensor de cualquier habitación cuando es de noche. La persona mayor debe estar sola en casa para que se registre este primer movimiento. Las reglas que forman este módulo, ordenadas por orden en el que se van deduciendo las cosas, son las siguientes:

Regla 1

```
(defrule Funcionalidad3
        ?f <- (modulo persona_despierta_noche)
        (ultimo_registro movimiento ?h ?instante)
        ?f2 <- (primer_movimiento_noche ?i)
        (test (> (- ?instante ?i) 900))
        (numero_de_personas ?numero)

        (hora_actual ?h1)
        (minutos_actual ?m1)
        (segundos_actual ?s1)
        =>
        (if (and (< ?instante ?*hora_inicio_noche_persona_despierta_noche*) (> ?instante
?*hora_fin_noche_persona_despierta_noche*) (eq ?numero 1))
        then
                (printout t ?h1 ":" ?m1 ":" ?s1 ": La persona lleva más de 15 minutos
despierta" crlf)
        )

        (retract ?f)
        (retract ?f2)
)
```

Esta regla comprueba que el primer movimiento de la noche se hizo hace más de 15 minutos (900 segundos). Para ello, se calcula la diferencia entre el último registro de movimiento del sensor de cualquier habitación y el primer movimiento de la noche. Si la diferencia es mayor de 15 minutos, significa que la persona mayor lleva más de 15 minutos despierta.

Regla 2

```
(defrule BorrarFuncionalidad3
        ?f <- (modulo persona_despierta_noche)
        ?f2 <- (primer_movimiento_noche ?i)
        (not (estado_habitacion (estado activa)))

        ;; Estos hechos son para una salida en pantalla más organizada
        (hora_actual ?h1)
        (minutos_actual ?m1)
        (segundos_actual ?s1)
        =>
        (printout t ?h1 ":" ?m1 ":" ?s1 ": La persona está dormida" crlf)
        (retract ?f)
        (retract ?f2)
)
```

Si el módulo está activo y todas las habitaciones están inactivas, se deduce que la persona se ha ido a dormir y se elimina el primer movimiento que hizo en la noche.

Regla 3

```
(defrule BorrarPrimerMovimientoNoche
        (primer_movimiento_noche ?i)
        (not (primer_movimiento_noche ?value2&:(minimo ?value2 ?i)))
        ?borrar1 <- (primer_movimiento_noche ?instante)
        (test (< ?i ?instante))
        =>
        (retract ?borrar1)
)
```

Cada vez que se active un sensor de movimiento durante la noche se va a insertar el hecho (primer_movimiento_noche ?i) en la base de hechos, pero sólo es interesante mantener el primer registro de movimiento cuando es de noche. Esta regla elimina los hechos (primer_movimiento_noche ?i) de forma que sólo deja en la base de hechos el primer movimiento con la hora más temprana.

Regla 4

```
(defrule BorrarPrimerMovimientoNoche2
        ?f <- (primer_movimiento_noche ?i)
        (HoraActualizada ?horaactual)

        (test (and (< ?horaactual ?*hora_inicio_noche_persona_despierta_noche*) (>
?horaactual ?*hora_fin_noche_persona_despierta_noche*)))
        =>
        (retract ?f)
)
```

Esta regla borra de la base de hechos el (primer_movimiento_noche ?i) cuando la hora del sistema no está en el intervalo de noche.

Validación

Suponiendo que la persona ya está dormida y que está sola en casa:

```
(datosensor 01 00 0 movimiento dormitorio on)
(datosensor 01 00 5 movimiento pasillo on)
```

Si a los 15 minutos alguna habitación está activa, el sistema alertaría de que la persona lleva más de 15 minutos despierta.

Suponiendo que la persona ya está dormida y que está sola en casa:

```
(datosensor 01 00 0 movimiento dormitorio on)
(datosensor 01 00 5 movimiento pasillo on)
(datosensor 01 01 5 movimiento pasillo off)
(datosensor 01 01 5 movimiento dormitorio off)
```

En este caso, el sistema detectaría que la persona se ha ido a dormir.

Módulo 4: Estando sola, la persona lleva más de 20 minutos en el baño

Este módulo es similar al anterior. Se activa cuando se registra un movimiento de la persona (estando sola en casa) en el baño. Para este módulo se utiliza un nuevo hecho llamado (instante_entra_banio ?instante), que mantiene en la base de hechos la hora en la que la persona entró al baño. La persona mayor debe estar sola en casa para que se registre este primer movimiento. Las reglas que forman este módulo, ordenadas por orden en el que se van deduciendo las cosas, son las siguientes:

Regla 1

```
((defrule Funcionalidad4
        ?f <- (modulo persona_mucho_tiempo_banio)
        (HoraActualizada ?hora)
        ?f2 <- (instante_entra_banio ?instante)
        (test (> (- ?hora ?instante) 1200))

        (hora_actual ?h1)
        (minutos_actual ?m1)
        (segundos_actual ?s1)
        =>
        (printout t ?h1 ":" ?m1 ":" ?s1 ": La persona lleva más de 20 minutos en el banio"
crlf)
        (retract ?f)
        (retract ?f2)
)
```

Esta regla comprueba si el instante en el que entró la persona mayor al baño y la hora actual del sistema tienen una diferencia de más de 20 minutos (1200 segundos).

Regla 2

```
(defrule BorrarFuncionalidad4
        (declare (salience 1000))
        ?f <- (modulo persona_mucho_tiempo_banio)
        ?f2 <- (instante_entra_banio ?i)
        (ultimo_registro movimiento ?h ?instante)
        (posible_pasar ?h banio)
        (test (neq ?h banio))
        (test (> ?instante ?i))

        (hora_actual ?h1)
        (minutos_actual ?m1)
        (segundos_actual ?s1)
        =>
        (printout t ?h1 ":" ?m1 ":" ?s1 ": La persona ha salido del banio" crlf)
        (retract ?f)
        (retract ?f2)
)
```

Si el módulo está activo (la persona está en el baño) y se enciende una habitación que conecta con el baño, deduzco que la persona ha salido del baño y elimino el hecho (instante_entra_banio ?i).

Regla 3

```
(defrule BorrarInstanteEntradaBanio
        (instante_entra_banio ?i)
        (not (instante_entra_banio ?value2&:(minimo ?value2 ?i)))
        ?borrar1 <- (instante_entra_banio ?instante)
        (test (< ?i ?instante))
        =>
        (retract ?borrar1)
)
```

Cada vez que se active el sensor de movimiento del baño se va a insertar el hecho (instante_entra_banio ?i) en la base de hechos, pero sólo es interesante mantener el primer instante en el que entró al baño. Esta regla elimina los hechos (nstante_entra_banio ?i) de forma que sólo deja en la base de hechos el primer instante en el que se activó el sensor del baño.

Validación

Suponiendo que la persona está sola en casa:

```
(datosensor 16 00 0 movimiento banio on)
```

Si a los 20 minutos la persona no ha salido del baño, el sistema alertaría de ello.

Suponiendo que la persona está sola en casa:

```
(datosensor 16 00 0 movimiento banio on)
(datosensor 16 05 0 movimiento pasillo on)
(datosensor 16 06 0 movimiento banio off)
```

En este caso el sistema detectaría que la persona ha salido del baño.

Módulo 5: La persona no ha ido al baño en las últimas 12 horas

Este módulo se activa en cada iteración (segundo) del simulador. Es bastante sencillo y consta únicamente de una regla:

Regla 1

```
(defrule Funcionalidad5
        ?f <- (modulo persona_no_ha_ido_banio)
        (HoraActualizada ?hora)
        (ultimo_registro movimiento banio ?i)
        (test (> (- ?hora ?i) 43200))

        (hora_actual ?h1)
        (minutos_actual ?m1)
        (segundos_actual ?s1)

        =>
        (printout t ?h1 ":" ?m1 ":" ?s1 ": La persona lleva sin ir al banio más de 12 horas"
crlf)
        (retract ?f)
)
```

Esta regla comprueba cuándo se activó por última vez el sensor de movimiento del baño. Si el último registro del baño fue hace más de 12 horas (43200 segundos), significa que la persona lleva sin ir al baño más de 12 horas.

Validación

Esta validación no se puede simular como tal. Simplemente, si no aparece un hecho del tipo (datosensor HH MM SS movimiento banio on) en las últimas 12 horas, el sistema alertaría de ello.

Módulo 6: La persona ha ido varias veces al baño en las últimas 3 horas

Este módulo se activa en cada iteración (segundo) del simulador. La idea de este módulo es contar las veces que se ha activado el baño en las últimas 3 horas cuando la persona está sola en casa. Para ello, cada vez que el sensor del baño se desactive y la persona esté sola en casa (la persona habrá salido el baño), se registra en la base de hechos (persona_entra_banio ?instante). Para ello se utiliza la variable global *n_neces_banio*. Las reglas son las siguientes:

Regla 1

```
(defrule Funcionalidad6
        (declare (salience 1000))
        ?f <- (modulo persona_varias_veces_banio)

        =>

        (assert (ContarHechos))
        (retract ?f)
)
```

Esta regla activa el contador con el hecho (ContarHechos).

Regla 2

```
(defrule Contador6
        (declare (salience 999))
        (ContarHechos)

        (HoraActualizada ?i)
        (persona_entra_banio ?hora)
        (test (< (- ?i ?hora) 10800))

        =>

        (bind ?*n_veces_banio* (+ ?*n_veces_banio* 1))
)
```

Esta regla cuenta los hechos (persona_entra_banio ?hora) cuyo instante sea las últimas 3 horas (10800 segundos). La variable global *n_veces_banio* va incrementando el contador en 1 en cada hecho.

Regla 3

```
(defrule ResultadoPrimero
        (declare (salience 10))

        (not (NumeroVecesBanio ?))

        ?eliminar1 <- (ContarHechos)

        =>

        (assert (NumeroVecesBanio ?*n_veces_banio*))
        (retract ?eliminar1)
        (bind ?*n_veces_banio* 0)
)
```

Una vez que se han contado todos los hechos, si en la base de hechos no hay ningún hecho del tipo (NumeroVecesBanio ?n) se entra en esta regla. Esta regla inserta en la base de hechos el número de veces que la persona mayor ha ido al baño en las últimas 3 horas y actualiza el contador *n_veces_banio* a 0.

Regla 4

```
(defrule Resultado6
        (declare (salience 10))

        ?eliminar1 <- (NumeroVecesBanio ?)

        ?eliminar2 <- (ContarHechos)

        =>

        (retract ?eliminar1)
        (retract ?eliminar2)

        (assert (NumeroVecesBanio ?*n_veces_banio*))

        (bind ?*n_veces_banio* 0)
)
```

Regla igual que la anterior pero cuando ya existe en la base de hechos (NumeroVecesBanio ?n).

Regla 5

```
(defrule BorrarPrimeraEntradaBanio

        (HoraActualizada ?i)
        ?f <- (persona_entra_banio ?hora)
        (test (> (- ?i ?hora) 10800))
        =>
        (retract ?f)
)
```

Esta regla borra los hechos (persona_entra_banio ?i) que hayan pasado hace más de 3 horas, de forma que no se tengan en cuenta en el contador.

Regla 6

```
(defrule DeducirPersonaVariasVecesBanio
        (declare (salience -10))

        (NumeroVecesBanio ?n)
        (test (> ?n 5))
        =>
        (printout t "La persona ha ido varias veces al banio en las ultimas 3 horas" crlf)
)
```

Esta regla es la que deduce. Si el número de veces del baño en el hecho (NumeroVecesBanio ?n) es mayor que 5, deduzco que la persona ha ido varias veces al baño en las últimas 3 horas.

Validación

Suponiendo que la persona está sola en casa:

```
(datosensor 11 00 0 movimiento banio on)
(datosensor 11 01 0 movimiento banio off)
(datosensor 11 01 01 movimiento pasillo on)
(datosensor 11 15 0 movimiento banio on)
(datosensor 11 16 0 movimiento banio off)
(datosensor 11 30 0 movimiento banio on)
(datosensor 11 31 0 movimiento banio off)
```

En este caso, se detecta que la persona ha ido 3 veces al baño.

Módulo 7: La asistenta llega tarde

Este módulo se activa en cada iteración (segundo) del simulador. Es un módulo bastante sencillo que consta únicamente de 1 regla:

Regla 1

```
(defrule Funcionalidad7
        (declare (salience -1000))
        ?f <- (modulo asistenta_llega_tarde)
        (HoraActualizada ?h)
        (test (eq ?h (+ ?*hora_fin_llegada_asistenta* 1)) )
        (asistenta_en_casa false)
        (dia_de_la_semana ?d)

        (hora_actual ?h1)
        (minutos_actual ?m1)
        (segundos_actual ?s1)
        =>

        (if (neq ?d domingo) then
                (printout t ?h1 ":" ?m1 ":" ?s1 ": La asistenta llega tarde" crlf)
        )

        (retract ?f)
)
```

Esta regla comprueba si la hora actual del sistema es mayor que la hora límite de llegada de la asistenta. Si la asistenta no ha llegado y la hora es mayor que la hora indicada en *hora_fin_llegada_asistenta*, deduzco que la asistenta llega tarde. Además, se tiene en cuenta que para avisar de esto el día tiene que ser distinto al domingo, pues los domingos la asistenta no llega a casa.

Validación

En el primer módulo se detectaba cuando una persona entra a la casa. Si no se detecta que alguien entra a la casa entre la hora a la que supuestamente debe llegar la asistenta, significaría que la asistenta llega tarde y el sistema alertaría de ello.

Módulo 8: Es tarde y la persona no se ha ido a dormir.

Este módulo se activa en cada iteración (segundo) del simulador. Es un módulo bastante sencillo que consta únicamente de 1 regla:

Regla 1

```
(defrule Funcionalidad8
        (declare (salience -1000))
        ?f <- (modulo persona_no_se_ha_dormido)
        (HoraActualizada ?hora)
        (not (estado_habitacion (estado activa)))

        (ultima_activacion movimiento ?hab ?instante)
        (not (ultima_activacion movimiento ?hab ?value2&:(maximo ?value2 ?instante)))

        (test (neq ?hab dormitorio))

        (numero_de_personas ?n)

        (hora_actual ?h1)
        (minutos_actual ?m1)
        (segundos_actual ?s1)
        =>
        (if (and (> ?hora ?*hora_inicio_persona_durmiendo_tarde*) (< ?instante
?*hora_fin_persona_durmiendo_tarde*) (eq ?n 1))
        then
                (printout t ?h1 ":" ?m1 ":" ?s1 ": Es tarde y la persona no se ha ido a dormir"
crlf)
        )
        (retract ?f)
)
```

Esta regla primero comprueba si no hay ninguna habitación activa. Si no hay ninguna habitación activa, es porque la persona estará durmiendo. A continuación se mira la habitación que se activó por última vez. Si la habitación que se activó por última vez **no** es el dormitorio y la hora en la que se activó está en el intervalo de *hora_inicio_persona_durmiendo_tarde* y *hora_fin_persona_durmiendo_tarde*, deduzco que la persona se fue a dormir tarde.

Validación

No se puede hacer una situación expresamente que detecte este módulo. La validación es bastante simple: si es tarde y hay alguna habitación activa, significa que la persona sigue despierta, con lo cual el sistema alertaría de ello.

Módulo 9: La persona tiene menos actividad de lo normal

Este módulo se activa en cada iteración (segundo) del simulador. La idea de este módulo es contar los pasos de una habitación a otra que se han producido desde las 00:00 hasta la hora actual. Recordar que los pasos se registran en el template (se_ha_producido_paso (instante ?hora)) de acuerdo al conocimiento del profesor en la práctica 1. Los pasos se cuentan en la variable global *n_pasos*. Las reglas son muy similares al contador que se utilizó en el módulo 6:

Regla 1

```
(defrule Funcionalidad9
        (declare (salience 1000))
        ?f <- (modulo persona_inactiva)

        =>

        (assert (ContarHechosPersonaInactiva))
        (retract ?f)
)
```

Esta regla activa el contador de pasos.

Regla 2

```
(defrule Contador9
        (declare (salience 999))
        (ContarHechosPersonaInactiva)

        (HoraActualizada ?i)
        (se_ha_producido_paso (instante ?hora))
        (test (< (- ?i ?hora) 43200))

        =>

        (bind ?*n_pasos* (+ ?*n_pasos* 1))
)
```

Esta regla cuenta los hechos (se_ha_producido_paso (instante ?hora)) cuyo instante sea las últimas 12 horas (43200 segundos). La variable global *n_pasos* va incrementando el contador en 1 en cada hecho.

Regla 3

```
(defrule ResultadoPrimero9
        (declare (salience 10))

        (not (NumeroPasos ?))

        ?eliminar1 <- (ContarHechosPersonaInactiva)

        =>

        (assert (NumeroPasos ?*n_pasos*))
        (retract ?eliminar1)
        (bind ?*n_pasos* 0)
)
```

Una vez que se han contado todos los hechos, si en la base de hechos no hay ningún hecho del tipo (NumeroPasos ?n) se entra en esta regla. Esta regla inserta en la base de hechos el número de pasos que se han contado y actualiza el contador *n_pasos* a 0.

Regla 4

```
(defrule Resultado9
        (declare (salience 10))

        ?eliminar1 <- (NumeroPasos ?)

        ?eliminar2 <- (ContarHechosPersonaInactiva)

        =>

        (retract ?eliminar1)
        (retract ?eliminar2)

        (assert (NumeroPasos ?*n_pasos*))

        (bind ?*n_pasos* 0)
)
```

Regla igual que la anterior pero cuando ya existe en la base de hechos (NumeroPasos ?n).

Regla 5

```
(defrule BorrarPasos

        (HoraActualizada ?i)
        ?f <- (se_ha_producido_paso (instante ?hora))
        (test (> (- ?i ?hora) 43200))
        =>
        (retract ?f)
)
```

Esta regla elimina los pasos que se han producido hace más de 12 horas, pues ya no harán falta para contarlos.

Regla 6

```
(defrule DeducirPersonaInactiva
        (declare (salience -10))
        (HoraActualizada ?i)
        (NumeroPasosNormales ?n_pasos ?hora)
        (test (eq ?i ?hora))

        (NumeroPasos ?n)
        (test (< ?n ?n_pasos))

        ;; Estos hechos son para una salida en pantalla más organizada
        (hora_actual ?h1)
        (minutos_actual ?m1)
        (segundos_actual ?s1)
        =>
        (printout t ?h1 ":" ?m1 ":" ?s1 ": La persona tiene menos actividad de lo normal" crlf)
)
```

Esta regla es la que deduce si la persona tiene menos actividad de lo normal. Para ello, en la base de hechos hay hechos del tipo (NumeroPasosNormales ?n_pasos ?hora) que significa que el número de pasos normales que da la persona normal desde las 00:00 hasta ?hora es ?n_pasos. Este número de pasos se podría calcular de forma más compleja utilizando aprendizaje automático. Lo importante de esta regla es que si la hora actual del sistema coincide con ?hora, se compara el número de pasos que se han contado con ?n_pasos, y si son menos pasos, significa que la persona no ha realizado tantos desplazamientos como hace normalmente y, por tanto, tiene menos actividad de lo normal.

Validación

Suponiendo que la persona está sola en casa y que es domingo:

```
(datosensor 10 00 0 puerta entrada calle off)
(datosensor 10 00 3 movimiento entrada on)
(datosensor 10 01 3 movimiento entrada off)
(datosensor 11 00 0 movimiento banio on)
(datosensor 11 01 0 movimiento banio off)
(datosensor 11 01 01 movimiento pasillo on)
(datosensor 11 15 0 movimiento banio on)
(datosensor 11 16 0 movimiento banio off)
(datosensor 11 30 0 movimiento banio on)
(datosensor 11 31 0 movimiento banio off)
```

Se detectarían 7 pasos en este caso. Si consideramos que dar 7 pasos durante esa franja horaria es poco, el sistem alertaría de ello.

Módulo 10: La persona hace más desplazamientos de lo normal en una franja horaria

Este módulo se activa en cada iteración (segundo) del simulador. La idea de este módulo es similar al módulo anterior: contar los pasos de una habitación a otra que se han producido pero esta vez en una franja horaria. Los pasos se cuentan en las variables *n_pasos_franja_maniana*, *n_pasos_franja_tarde,* y *n_pasos_franja_noche*, dependiendo de la franja en la que esté el simulador.

Regla 1

```
(defrule Funcionalidad10
        (declare (salience 1000))
        ?f <- (modulo persona_realiza_mas_desplazamientos)
        (HoraActualizada ?hora)
        =>
        (if (and (> ?hora ?*hora_inicio_maniana*) (< ?hora ?*hora_fin_maniana*)) then
                (assert (ContarHechosMasDesplazamientosFranjaManiana))

        else
                (if (and (> ?hora ?*hora_inicio_tarde*) (< ?hora ?*hora_fin_tarde*)) then
                (assert (ContarHechosMasDesplazamientosFranjaTarde))

                else
                (if (and (> ?hora ?*hora_inicio_noche*) (< ?hora ?*hora_fin_noche*)) then
                        (assert (ContarHechosMasDesplazamientosFranjaNoche))

                )
                )
        )

        (retract ?f)
)
```

Esta regla activa el contador de pasos. Dependiendo de la franja horaria en la que esté el simulador, activará el contador de *n_pasos_franja_maniana*, *n_pasos_franja_tarde,* o *n_pasos_franja_noche*.

Regla 2

```
(defrule ContadorFranjaManiana10
        (declare (salience 999))
        (ContarHechosMasDesplazamientosFranjaManiana)

        (se_ha_producido_paso (instante ?hora))

        =>

        (bind ?*n_pasos_franja_maniana* (+ ?*n_pasos_franja_maniana* 1))
)
```

Esta regla cuenta los hechos (se_ha_producido_paso (instante ?hora)) que hay en la base de hechos. La variable global *n_pasos_franja_maniana* va incrementando el contador en 1 en cada hecho.

Regla 3

```
(defrule ContadorFranjaTarde10
        (declare (salience 999))
        (ContarHechosMasDesplazamientosFranjaTarde)

        (se_ha_producido_paso (instante ?hora))

        =>

        (bind ?*n_pasos_franja_tarde* (+ ?*n_pasos_franja_tarde* 1))
)
```

Regla con el mismo objetivo que la anterior, pero contando los hechos cuando el simulador está en la franja de tarde.

Regla 4

```
(defrule ContadorFranjaNoche10
        (declare (salience 999))
        (ContarHechosMasDesplazamientosFranjaNoche)

        (se_ha_producido_paso (instante ?hora))

        =>

        (bind ?*n_pasos_franja_noche* (+ ?*n_pasos_franja_noche* 1))
)
```

Regla con el mismo objetivo que la anterior, pero contando los hechos cuando el simulador está en la franja de noche.

Regla 5

```
(defrule ResultadoPrimeroFranjaManiana
        (declare (salience 10))

        (not (NumeroPasosManiana ?))

        ?eliminar1 <- (ContarHechosMasDesplazamientosFranjaManiana)

        =>

        (assert (NumeroPasosManiana ?*n_pasos_franja_maniana*))
        (retract ?eliminar1)
        (bind ?*n_pasos_franja_maniana* 0)
)
```

Una vez que se han contado todos los hechos, si el simulador está en la franja de mañana y en la base de hechos no hay ningún hecho del tipo (NumeroPasosManiana ?n) se entra en esta regla. Esta regla inserta en la base de hechos el número de pasos que se han contado y actualiza el contador *n_pasos_franja_maniana* a 0.

Regla 6

```
(defrule ResultadoFranjaManiana
        (declare (salience 10))

        ?eliminar1 <- (NumeroPasosManiana ?)

        ?eliminar2 <- (ContarHechosMasDesplazamientosFranjaManiana)

        =>

        (retract ?eliminar1)
        (retract ?eliminar2)

        (assert (NumeroPasosManiana ?*n_pasos_franja_maniana*))

        (bind ?*n_pasos_franja_maniana* 0)
)
```

Regla igual que la anterior pero cuando ya existe en la base de hechos (NumeroPasosManiana ?n).

Regla 7

```
(defrule ResultadoPrimeroFranjaTarde
        (declare (salience 10))

        (not (NumeroPasosTarde ?))

        ?eliminar1 <- (ContarHechosMasDesplazamientosFranjaTarde)

        =>

        (assert (NumeroPasosTarde ?*n_pasos_franja_tarde*))
        (retract ?eliminar1)
        (bind ?*n_pasos_franja_tarde* 0)
)
```

Una vez que se han contado todos los hechos, si el simulador está en la franja de tarde y en la base de hechos no hay ningún hecho del tipo (NumeroPasosTarde ?n) se entra en esta regla. Esta regla inserta en la base de hechos el número de pasos que se han contado y actualiza el contador *n_pasos_franja_tarde* a 0.

Regla 8

```
(defrule ResultadoFranjaTarde
        (declare (salience 10))

        ?eliminar1 <- (NumeroPasosTarde ?)

        ?eliminar2 <- (ContarHechosMasDesplazamientosFranjaTarde)

        =>

        (retract ?eliminar1)
        (retract ?eliminar2)

        (assert (NumeroPasosTarde ?*n_pasos_franja_tarde*))

        (bind ?*n_pasos_franja_tarde* 0)
)
```

Regla igual que la anterior pero cuando ya existe en la base de hechos (NumeroPasosTarde ?n).

Regla 9

```
(defrule ResultadoPrimeroFranjaNoche
        (declare (salience 10))

        (not (NumeroPasosNoche ?))

        ?eliminar1 <- (ContarHechosMasDesplazamientosFranjaNoche)

        =>

        (assert (NumeroPasosNoche ?*n_pasos_franja_noche*))
        (retract ?eliminar1)
        (bind ?*n_pasos_franja_noche* 0)
)
```

Una vez que se han contado todos los hechos, si el simulador está en la franja de noche y en la base de hechos no hay ningún hecho del tipo (NumeroPasosNoche ?n) se entra en esta regla. Esta regla inserta en la base de hechos el número de pasos que se han contado y actualiza el contador *n_pasos_franja_noche* a 0.

Regla 10

```
(defrule ResultadoFranjaNoche
        (declare (salience 10))

        ?eliminar1 <- (NumeroPasosNoche ?)

        ?eliminar2 <- (ContarHechosMasDesplazamientosFranjaNoche)

        =>

        (retract ?eliminar1)
        (retract ?eliminar2)

        (assert (NumeroPasosNoche ?*n_pasos_franja_noche*))

        (bind ?*n_pasos_franja_noche* 0)
)
```

Regla igual que la anterior pero cuando ya existe en la base de hechos (NumeroPasosNoche ?n).

Regla 11

```
(defrule DeducirPersonaMasDesplazamientosFranjaManiana
        (declare (salience -10))

        (NumeroPasosNormalesFranja ?n_pasos franja_maniana)
        (NumeroPasosManiana ?n)
        (test (> ?n (+ ?n_pasos 10)))

        ;; Estos hechos son para una salida en pantalla más organizada
        (hora_actual ?h1)
        (minutos_actual ?m1)
        (segundos_actual ?s1)
        =>
        (printout t ?h1 ":" ?m1 ":" ?s1 ": La persona realiza más desplazamientos en la franja
de maniana" crlf)
)
```

Esta regla es la que deduce si la persona tiene realiza más desplazamientos de lo normal en la franja de mañana. Para ello, en la base de hechos hay hechos del tipo (NumeroPasosNormalesFranja ?n_pasos franja_maniana) que significa que el número de pasos normales que da la persona normal en la franja de mañana es ?n_pasos. Este número de pasos se podría calcular de forma más compleja utilizando aprendizaje automático. Lo importante de esta regla es que se compara el número de pasos que se han contado con ?n_pasos+10 (+10 para dar un poco de margen), y si son más pasos, significa que la persona ha realizado desplazamientos que normalmente no hace por la mañana.

Regla 12

```
(defrule DeducirPersonaMasDesplazamientosFranjaTarde
        (declare (salience -10))

        (NumeroPasosNormalesFranja ?n_pasos franja_tarde)
        (NumeroPasosTarde ?n)
        (test (> ?n (+ ?n_pasos 10)))

        ;; Estos hechos son para una salida en pantalla más organizada
        (hora_actual ?h1)
        (minutos_actual ?m1)
        (segundos_actual ?s1)
        =>
        (printout t ?h1 ":" ?m1 ":" ?s1 ": La persona realiza más desplazamientos en la franja
de tarde" crlf)
)
```

Mismo objetivo que la regla anterior, pero para la franja de por la tarde.

Regla 13

```
(defrule DeducirPersonaMasDesplazamientosFranjaTarde
        (declare (salience -10))

        (NumeroPasosNormalesFranja ?n_pasos franja_tarde)
        (NumeroPasosTarde ?n)
        (test (> ?n (+ ?n_pasos 10)))

        ;; Estos hechos son para una salida en pantalla más organizada
        (hora_actual ?h1)
        (minutos_actual ?m1)
        (segundos_actual ?s1)
        =>
        (printout t ?h1 ":" ?m1 ":" ?s1 ": La persona realiza más desplazamientos en la franja
de tarde" crlf)
)
```

Mismo objetivo que la regla anterior, pero para la franja de por la noche.

Validación

La validación de este módulo es similar a la del módulo anterior, pero esta vez se compararían las franjas horarias (mañana, tarde y noche).

Manual de uso del sistema

Para usar el sistema, se introduce en el fichero *DatosSimulados.txt* la simulación de los sensores que se desee. En el fichero *SituacionInicial.txt* se puede cambiar la hora de inicio del simulador (por defecto, las 10:00:00). A continuación, se introduce en CLIPS la orden (batch simulacion.bat) y se pedirá al usuario que se introduzca el número de segundos que queremos que transcurran en el simulador. Por pantalla aparecerán mensajes cada vez que se active/desactive un sensor y cuando se detecten las anomalías de acuerdo a los módulos presentados anteriormente. Finalmente, se guardarán todos los hechos de cómo ha quedado la casa en la última iteración en el fichero *Situacionfinal.txt*.