2012

Implementación del prototipo

Tercer trabajo de Ingeniería del Conocimiento

Alberto Mireles Suárez
David Morales Sáez

ULPGC
21/05/2012

Índice

Introducción 2

Alcance de la implementación 2

Ejecución 3

Código 5

Atributos 5

Declaraciones 6

Hechos 11

Previo_inicio 11

Inicialización 13

Avanza 14

Constraints 17

Preferencias 21

Ordenación 22

Introducción

En este trabajo se detalla el alcance de la implementación del prototipo que asista en la elaboración de horarios de un Colegio de Educación Primaria situado en el Ejido, Telde.

Alcance de la implementación

El prototipo del Sistema Experto está basado en el conocimiento del dominio del trabajo anterior, por lo que se ha respetado el dominio usando los conceptos del modelo.

Con el fin de generar las soluciones, primero se han de introducir los profesores, las aulas y los grupos definiendo instancias de las clases correspondientes. En este caso se han reflejado tres aulas que representan tres grupos de 5º de primaria. Entonces, se generan las tripletas de [profesor, asignatura, aula] y vuelta atrás empieza a aplicar las restricciones del modelo. Las restricciones que se han implementado son las siguientes:

- Una asignatura debe ser impartida sólo por los profesores indicados. Esto implica que la asignatura sólo puede impartirse por un profesor si está definida una instancia de "imparte" entre el profesor y la asignatura.

- En un aula sólo se pueden impartir asignaturas en las que dicho aula se requiera. Esto implica que una asignatura solo puede impartirse si hay una instancia de la relación "requiere", indicando que una asignatura requiere un aula específica.

- Una asignatura solo puede aparecer en un mismo día un máximo de dos veces, es decir, al día se puede dar un máximo de dos sesiones de la misma asignatura.

- En un aula sólo se pueden impartir asignaturas que sean de esa aula, y no de otra.

- Un profesor no puede estar en múltiples aulas al mismo tiempo, por tanto, no puede figurar en la misma celda en dos o más aulas.

- No se puede rellenar un horario de un aula que ya ha sido rellenado. Para esta restricción se han empleado tres reglas.

El resto de las restricciones que se indican en el anterior trabajo no han sido implementadas ya que este conjunto es representativo para ilustrar la traducción de reglas al entorno del ART*Enterprise y el funcionamiento de las restricciones.

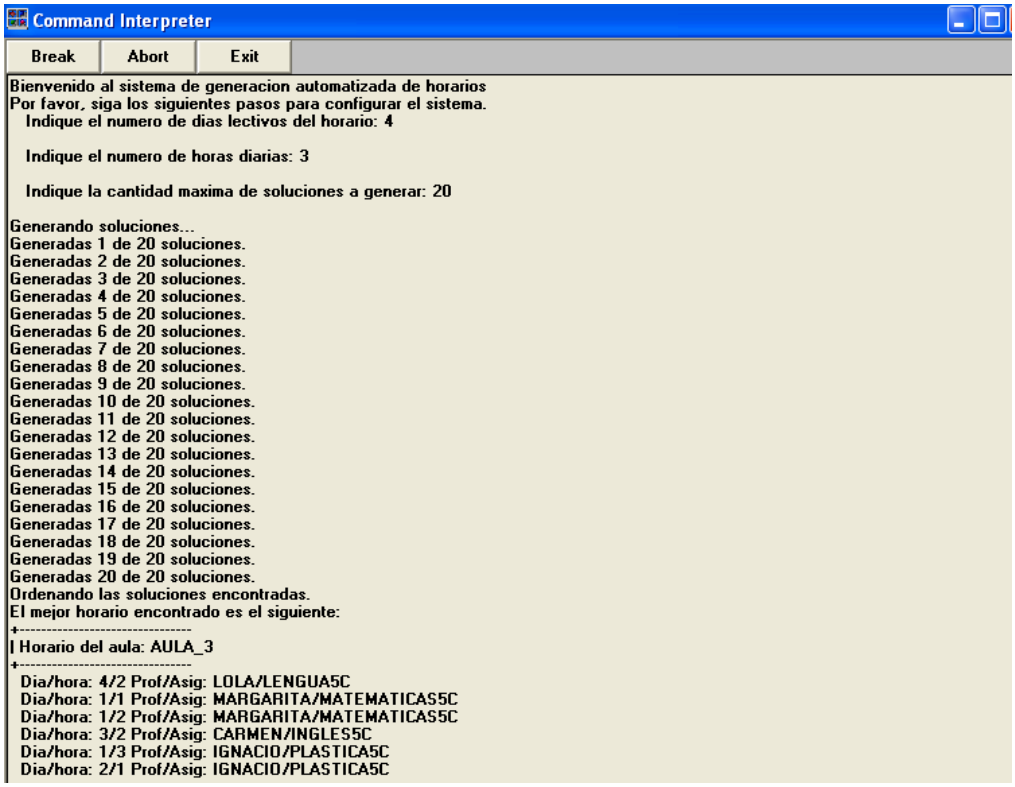
Una solución válida se corresponde con un conjunto de horarios que representa todos los horarios de todas las aulas. A su vez, un horario es un agregado de las celdas que lo componen. Esto implica que a medida que vamos generando cada una de las soluciones, hemos de generar las instancias de todos los objetos de las clases CeldaHorario, Horario y ConjuntoHorarios. Así seremos capaces de acceder al atributo de puntuación de cada conjunto de horarios y ser capaces de compararlos en base a las preferencias.

Una vez se generan las soluciones, se asigna una puntuación a cada solución dependiendo de si es mejor en base a las preferencias. La implementación de las preferencias se hace con una regla, la cual calcula la puntuación de cada preferencia con una función para cada una, para luego sumar las puntuaciones. Tras este paso se ejecutarán las reglas de ordenación de las soluciones en base a su valor.

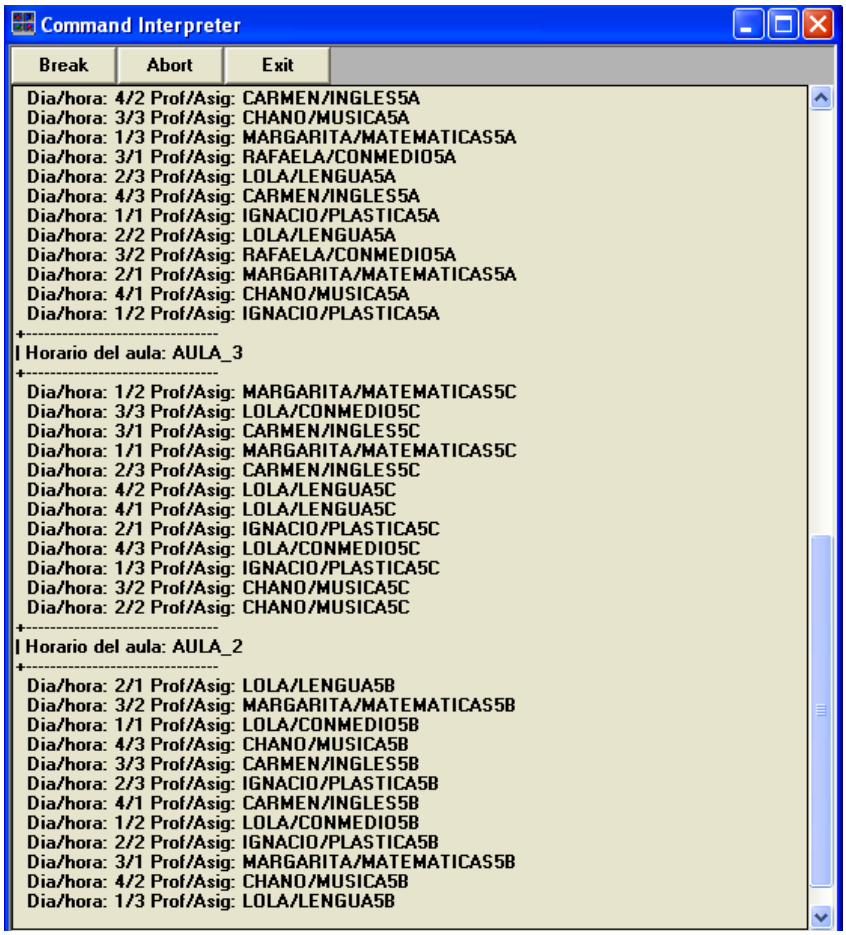
Para ilustrar el uso de las preferencias se han implementado dos preferencias: una que asigna un mayor valor a aquellas soluciones en las que las asignaturas de lengua y matemáticas sean impartidas en las tres primeras horas, y otra en la que se premian los horarios en los que se impartan horas seguidas de las asignaturas de lengua y matemáticas.

Ejecución

Antes de la generación de los dominios para posteriormente generar las soluciones, se solicita al usuario cierta información sobre el tamaño de los horarios: el número de días y el número de horas por día. A su vez, solicita el número máximo de soluciones que se quiere generar, de esta manera se simplifica la tarea de realizar pruebas sin tener que modificar el fichero fuente.
Durante la ejecución se irá mostrando un mensaje cada vez que se genere una solución y finalmente, tras ordenar las soluciones mostrará aquel con mejor puntuación. A continuación se muestra una captura del intérprete de comandos:



Como ejemplo, a continuación se muestra la captura que muestra el horario con mayor puntuación tras evaluar 25 soluciones:



Código

Atributos

```
(define-attribute grupo-m relation
    (cardinality multiple))

(define-attribute horario-m relation
    (cardinality multiple))

(define-attribute profesor-m relation
    (cardinality multiple))

(define-attribute aula-m relation
    (cardinality multiple))

(define-attribute asignatura-m relation
    (cardinality multiple))

(define-class ConjuntoHorarios nil
    (nombre)
    (grupo-m)
    (horario-m)
    (profesor-m)
    (aula-m)
    (asignatura-m)
    (puntuacion 0)

(define-attribute CeldaHorario-rel relation
    (cardinality multiple))

(define-class horario nil
    (IdAula)
    (CeldaHorario-rel))

(define-class CeldaHorario nil
    (profesor-s)
    (asignatura-s)
    (aula-s)
    (fecha)
    (dia)
    (hora))

(define-class grupo nil
    (NombreGrupo) )

(define-class aula nil
    (numero))

(define-class asignada-a nil
    (aula-s)
    (grupo-s))

(define-class asignatura nil
    (nombre)
    (NumGrupo)
    (NumHorasClase 0))
```

```
(define-class requiere nil
    (asignatura-s)
    (aula-s))

(define-class profesor nil
    (nombre))

(define-class imparte nil
    (profesor-s)
    (asignatura-s))
```

Declaraciones

```
(define-instance Aula_1 aula
    (numero 1))

(define-instance Aula_2 aula
    (numero 2))

(define-instance Aula_3 aula
    (numero 3))

(define-instance 5A grupo
    (NombreGrupo 5A))

(define-instance 5B grupo
    (NombreGrupo 5B))

(define-instance 5C grupo
    (NombreGrupo 5C))

(define-instance Aula_1-5A asignada-a
    (aula-s Aula_1)
    (grupo-s 5A))

(define-instance Aula_2-5B asignada-a
    (aula-s Aula_2)
    (grupo-s 5B))

(define-instance Aula_3-5C asignada-a
    (aula-s Aula_3)
    (grupo-s 5C))

(define-instance ConMedio5A asignatura
    (nombre Naturales)
    (NumGrupo 5A)
    (NumHorasClase 4))

(define-instance ConMedio5B asignatura
    (nombre Naturales)
    (NumGrupo 5B)
    (NumHorasClase 4))

(define-instance Plastica5A asignatura
    (nombre Plastica)
    (NumGrupo 5A)
    (NumHorasClase 4))
```

```
(define-instance Plastica5B asignatura
    (nombre Plastica)
    (NumGrupo 5B)
    (NumHorasClase 4))

(define-instance ConMedio5C asignatura
    (nombre Fisica)
    (NumGrupo 5C)
    (NumHorasClase 3))

(define-instance Plastica5C asignatura
    (nombre Quimica)
    (NumGrupo 5C)
    (NumHorasClase 3))

(define-instance Lengua5A asignatura
    (nombre Lengua)
    (NumGrupo 5A)
    (NumHorasClase 6))

(define-instance Lengua5B asignatura
    (nombre Lengua)
    (NumGrupo 5B)
    (NumHorasClase 6))

(define-instance Lengua5C asignatura
    (nombre Lengua)
    (NumGrupo 5C)
    (NumHorasClase 3))


(define-instance Matematicas5A asignatura
    (nombre Matematicas)
    (NumGrupo 5A)
    (NumHorasClase 5))

(define-instance Matematicas5B asignatura
    (nombre Matematicas)
    (NumGrupo 5B)
    (NumHorasClase 5))

(define-instance Matematicas5C asignatura
    (nombre Matematicas)
    (NumGrupo 5C)
    (NumHorasClase 3))

(define-instance Ingles5A asignatura
    (nombre Ingles)
    (NumGrupo 5A)
    (NumHorasClase 4))

(define-instance Ingles5B asignatura
    (nombre Ingles)
    (NumGrupo 5B)
    (NumHorasClase 4))

(define-instance Ingles5C asignatura
    (nombre Ingles)
    (NumGrupo 5C)
```

```
    (NumHorasClase 4))

(define-instance Musica5A asignatura
    (nombre Musica)
    (NumGrupo 5A)
    (NumHorasClase 2))

(define-instance Musica5B asignatura
    (nombre Musica)
    (NumGrupo 5B)
    (NumHorasClase 2))

(define-instance Musica5C asignatura
    (nombre Musica)
    (NumGrupo 5C)
    (NumHorasClase 2))

(define-instance Lengua5A-Aula_1 requiere
    (asignatura-s Lengua5A)
    (aula-s Aula_1))

(define-instance Lengua5B-Aula_2 requiere
    (asignatura-s Lengua5B)
    (aula-s Aula_2))

(define-instance Lengua5C-Aula_3 requiere
    (asignatura-s Lengua5C)
    (aula-s Aula_3))



(define-instance Matematicas5A-Aula_1 requiere
    (asignatura-s Matematicas5A)
    (aula-s Aula_1))

(define-instance Matematicas5B-Aula_2 requiere
    (asignatura-s Matematicas5B)
    (aula-s Aula_2))

(define-instance Matematicas5C-Aula_3 requiere
    (asignatura-s Matematicas5C)
    (aula-s Aula_3))

(define-instance ConMedio5A-Aula_1 requiere
    (asignatura-s ConMedio5A)
    (aula-s Aula_1))

(define-instance ConMedio5B-Aula_2 requiere
    (asignatura-s ConMedio5B)
    (aula-s Aula_2))

(define-instance Plastica5A-Aula_1 requiere
    (asignatura-s Plastica5A)
    (aula-s Aula_1))

(define-instance Plastica5B-Aula_2 requiere
    (asignatura-s Plastica5B)
    (aula-s Aula_2))
```

```
(define-instance ConMedio5C-Aula_3 requiere
    (asignatura-s ConMedio5C)
    (aula-s Aula_3))

(define-instance Plastica5C-Aula_3 requiere
    (asignatura-s Plastica5C)
    (aula-s Aula_3))

(define-instance Ingles5A-Aula_1 requiere
    (asignatura-s Ingles5A)
    (aula-s Aula_1))

(define-instance Ingles5B-Aula_2 requiere
    (asignatura-s Ingles5B)
    (aula-s Aula_2))

(define-instance Ingles5C-Aula_3 requiere
    (asignatura-s Ingles5C)
    (aula-s Aula_3))


(define-instance Musica5A-Aula_1 requiere
    (asignatura-s Musica5A)
    (aula-s Aula_1))

(define-instance Musica5B-Aula_2 requiere
    (asignatura-s Musica5B)
    (aula-s Aula_2))



(define-instance Musica5C-Aula_3 requiere
    (asignatura-s Musica5C)
    (aula-s Aula_3))

(define-instance LOLA profesor
    (nombre Lola))

(define-instance IGNACIO profesor
    (nombre Ignacio))

(define-instance MARGARITA profesor
    (nombre Margarita))

(define-instance RAFAELA profesor
    (nombre Rafaela))

(define-instance CARMEN profesor
    (nombre Carmen))

(define-instance CHANO profesor
    (nombre Chano))

(define-instance LOLA-imparte-Lengua5A imparte
    (profesor-s LOLA)
    (asignatura-s Lengua5A))

(define-instance LOLA-imparte-Lengua5B imparte
    (profesor-s LOLA)
```

```
    (asignatura-s Lengua5B))

(define-instance LOLA-imparte-Lengua5C imparte
    (profesor-s LOLA)
    (asignatura-s Lengua5C))

(define-instance MARGARITA-imparte-Matematicas5A imparte
    (profesor-s MARGARITA)
    (asignatura-s Matematicas5A))

(define-instance MARGARITA-imparte-Matematicas5B imparte
    (profesor-s MARGARITA)
    (asignatura-s Matematicas5B))

(define-instance MARGARITA-imparte-Matematicas5C imparte
    (profesor-s MARGARITA)
    (asignatura-s Matematicas5C))

(define-instance LOLA-imparte-ConMedio5C imparte
    (profesor-s LOLA)
    (asignatura-s ConMedio5C))

(define-instance RAFAELA-imparte-ConMedio5A imparte
    (profesor-s RAFAELA)
    (asignatura-s ConMedio5A))

(define-instance LOLA-imparte-ConMedio5B imparte
    (profesor-s LOLA)
    (asignatura-s ConMedio5B))


(define-instance IGNACIO-imparte-Plastica5A imparte
    (profesor-s IGNACIO)
    (asignatura-s Plastica5A))

(define-instance IGNACIO-imparte-Plastica5B imparte
    (profesor-s IGNACIO)
    (asignatura-s Plastica5B))

(define-instance IGNACIO-imparte-Plastica5C imparte
    (profesor-s IGNACIO)
    (asignatura-s Plastica5C))

(define-instance CARMEN-imparte-Ingles5A imparte
    (profesor-s CARMEN)
    (asignatura-s Ingles5A))

(define-instance CARMEN-imparte-Ingles5B imparte
    (profesor-s CARMEN)
    (asignatura-s Ingles5B))

(define-instance CARMEN-imparte-Ingles5C imparte
    (profesor-s CARMEN)
    (asignatura-s Ingles5C))

(define-instance CHANO-imparte-Musica5A imparte
    (profesor-s CHANO)
    (asignatura-s Musica5A))
```

```
(define-instance CHANO-imparte-Musica5B imparte
    (profesor-s CHANO)
    (asignatura-s Musica5B))

(define-instance CHANO-imparte-Musica5C imparte
    (profesor-s CHANO)
    (asignatura-s Musica5C))
```

Hechos

```
(define-fact-list datos
    (profesores)
    (asignaturas)
    (aulas)
    (grupos)
    (dominios)
    (solucion)
    (aulas-completas))

(define-global ?*NumSoluciones*      = 0  :reset)

(define-global ?*MaxSoluciones*    = 0 :reset)

(define-global ?*NumAulas*    = 3  :reset)

(define-global ?*NumHoras*    = 0  :reset)

(define-global ?*NumDias*     = 0  :reset)
```

Previo_inicio

```
(define-rule inicio
    (declare
        (ruleset previa-inicializacion)
        (salience 10000))
=>
    (enable-ruleset :all)
    (enable-ruleset previa-inicializacion)
    (disable-ruleset inicializacion)
    (disable-ruleset avances)
    (disable-ruleset constraints)
    (disable-ruleset preferencias)
    (disable-ruleset ordenacion)

        (printout t "Bienvenido al sistema de generacion automatizada de
horarios" t)
        (printout t "Por favor, siga los siguientes pasos para configurar el
sistema." t)
        (enable-ruleset :all)
        (disable-ruleset avances)
        (disable-ruleset preferencias)
        (printout t "   Indique el numero de dias lectivos del horario: " )
        (bind ?*NumDias* (read))
        (printout t "   Indique el numero de horas diarias: " )
        (bind ?*NumHoras* (read))
```

```
        (printout t "   Indique la cantidad maxima de soluciones a generar: "
)
        (bind ?*MaxSoluciones* (read))
        (bind ?x (* ?*NumAulas* ?*NumDias* ?*NumHoras*))
        (assert (estructura ?x))
        (printout t "Generando soluciones..." t)
)

(define-rule dame-profesores
    (declare
        (salience 9000)
        (ruleset previa-inicializacion))
    ?f<-(profesores $?prof)
    (object ?p
        (instance-of profesor))
    (test (not (member$ ?p ?prof)))
=>
    (assert (profesores $?prof ?p))
    (retract ?f))

(define-rule dame-aulas
    (declare
        (salience 9000)
        (ruleset previa-inicializacion))
    ?f<-(aulas $?aul)
    (object ?a
        (instance-of aula))
    (test (not (member$ ?a ?aul)))
=>
    (assert (aulas $?aul ?a))
    (retract ?f))


(define-rule dame-asignaturas
    (declare
        (salience 9000)
        (ruleset previa-inicializacion))
    ?f<-(asignaturas $?asig)
    (object ?a
        (instance-of asignatura))
    (test (not (member$ ?a ?asig)))
=>
    (assert (asignaturas $?asig ?a))
    (retract ?f))

(define-rule dame-grupos
    (declare
        (salience 9000)
        (ruleset previa-inicializacion))
    ?f<-(grupos $?grup)
    (object ?g
        (instance-of grupo))
    (test (not (member$ ?g ?grup)))
=>
    (assert (grupos $?grup ?g))
    (retract ?f))

(define-rule combina-aula-prof-asig
    (declare
```

```
        (salience 8000)
        (ruleset previa-inicializacion))
    ?f<-(dominios $?dom)
    ?f1<-(profesores $?Aula_1 ?b1 $?c1)
    ?f2<-(asignaturas $?Aula_2 ?b2 $?c2)
    ?f3<-(aulas $?Aula_3 ?b3 $?c3)
    (test (not (member$ (create$ ?b1 ?b2 ?b3) ?dom)))
=>
    (assert (dominios $?dom (?b1 ?b2 ?b3)))
    (retract ?f))

(define-rule genera-resto-dominios
    (declare
        (salience 7000)
        (ruleset previa-inicializacion))
    ?f<-(dominios $?a)
    (estructura ?x)
=>
    (for ?i from 1 to ?x do
        (assert (dominio ?i $?a)))
    (retract ?f))

(define-rule Salida-Previa-Ini
    (declare
        (ruleset previa-inicializacion)
        (salience -100))
=>
    (enable-ruleset inicializacion)
    (disable-ruleset previa-inicializacion))
```

Inicialización

```
(define-class elementos nil
    (posicion 0)
    (elemento 0)
    (eliminado 0))

(define-rule Genera-Elementos-Dominio
    (declare (ruleset inicializacion))
    (dominio ?pos $? ?ele $?)
=>
    (assert (object =(string-to-symbol
        (string-append "ele-"
            (sprintf "%d" ?pos) "-"
            (symbol-to-string (nth$ ?ele 1)) "-"
            (symbol-to-string (nth$ ?ele 2)) "-"
            (symbol-to-string (nth$ ?ele 3))))
        (instance-of elementos)
        (posicion ?pos)
        (elemento ?ele)
        (eliminado 0))))

(define-rule Salida-Inicializar
    (declare
```

```
        (ruleset inicializacion)
        (salience -100))
=>
    (enable-ruleset avances)
    (enable-ruleset constraints)
    (disable-ruleset inicializacion))
```

Avanza

```
; Incrementa en un nivel la solución parcial

(define-rule avanza
    (declare (salience 100) (ruleset avances))
    ?f<-(solucion $?a)
    (object ?
        (instance-of elementos)
        (elemento ?ele) ;buscamos un elemento
        (posicion ?b&:(= ?b (+ 1 (length$ ?a)))); del dominio siguiente
        (eliminado 0) ; que este disponible
    )
=>
    (assert (solucion $?a ?ele))
    (retract ?f)
)

; La aplicación acaba cuando tenemos el dominio 1 saturado

(define-rule Fin
    (declare (salience 3000) (ruleset avances))
    (not (object ?obj2
        (instance-of elementos)
        (posicion 1)
        (eliminado 0) ; que no tenga disponibles
    ))
=>
    (printout t "Se ha completado la generacion de soluciones" t)
    (halt)
)

; Detecta una solucion

(define-rule detecta-solucion
    (declare (salience 100) (ruleset avances))
    ?f<-(solucion $?a ?b)
    ?g<-(aulas-completas $?c)
    (estructura ?n&:(= (- ?n 1) (length$ ?a)))
    (object ?obj
        (posicion ?pos&:(= ?pos (+ 1 (length$ ?a))))
        (elemento ?b))
=>
    (bind ?*NumSoluciones* (+ 1 ?*NumSoluciones*))
    (printout t "Generadas " ?*NumSoluciones* " de " ?*MaxSoluciones* "
soluciones." t)
    (assert (sol-final ?*NumSoluciones* $?a ?b))
    (set-attribute-value ?obj eliminado 1)
```

```
    (retract ?g)
    (retract ?f)
    (assert (solucion $?a))
    (assert (aulas-completas))
)

; Restaura valores de dominios

(define-rule Restaura-nivel
    (declare (salience 400) (ruleset avances))
    (vuelta-atras)
    (solucion $?a)
    (object ?obj
        (instance-of elementos)
        (posicion ?b&:(= ?b (+ 1(length$ ?a))))
        (eliminado 1)
    )
=>
    (set-attribute-value ?obj eliminado 0)
)

; Detecta que tenemos un dominio saturado

(define-rule Detecta-DominioSaturado
    (declare (salience 300) (ruleset avances))
    (solucion $?sol)
    (object ?obj ;buscamos un objeto
        (instance-of elementos)
        (posicion ?b&:(= ?b (+ 1(length$ ?sol))));nos aseguramos que es un
dominio futuro
    )
    (not (object ?obj2
        (instance-of elementos)
        (posicion ?b)
        (eliminado 0) ; que no tenga disponibles
    ))
=>
    (assert (vuelta-atras))
)

; Anula el ultimo valor de la solución y elimina la condición de vuelta-atrás

(define-rule Fin-VueltaAtras
    (declare (salience 350) (ruleset avances))
    ?f<- (vuelta-atras)
    ?g<-(solucion $?inic ?a)
    (object ?obj; buscamos el objeto actual
        (instance-of elementos)
        (posicion ?b&:(= ?b (+ 1 (length$ ?inic)))) ; ultima posición
        (elemento ?a)
    )
=>
    (set-attribute-value ?obj eliminado 1)
    (assert (solucion $?inic))
    (retract ?f ?g)
)

; Genera la representación de la solución (CeldaHorario), cuando una solución
ha sido detectada
```

```
(define-rule genera-repr-solucion
    (declare
        (salience 101)
        (ruleset avances))
    ?f<-(sol-final ?numcurso $?celdas)
    (test (= (* ?*NumAulas* ?*NumDias* ?*NumHoras*) (length$ ?celdas)))
    (grupos      $?grups)
    (profesores  $?profs)
    (aulas       $?auls)
    (asignaturas $?asigs)
=>
    (bind ?pos 1)
    (bind ?horarios-curso
        (for ?naula from 1 to ?*NumAulas* collect$
            (bind ?celdas-horario
                (for ?ndia from 1 to ?*NumDias* collect$
                    (bind ?celdas-dia
                        (for ?nhora from 1 to ?*NumHoras* collect$
                            (bind ?celda (nth$ ?celdas ?pos)) ; saca asig-
prof-aula
                            (bind ?prof (nth$ ?celda 1))
                            (bind ?asig (nth$ ?celda 2))
                            (bind ?aul  (nth$ ?celda 3))
                            (bind ?pos (+ ?pos 1))
                            (bind ?fecha (+ ?ndia (/ (float ?nhora) 10)))
                            (assert (object =(string-to-symbol
                                                (string-append
                                                    "cel-"
                                                    (symbol-to-string ?prof)
                                                    "_"
                                                    (symbol-to-string ?asig)
                                                    "_"
                                                    (symbol-to-string ?aul)
                                                    "_"
                                                    (sprintf "dia%d" ?ndia)
                                                    "_"
                                                    (sprintf "hora%d" ?nhora)
                                                    "_"
                                                    (sprintf "curso%d"
?numcurso) ))
                                            (instance-of CeldaHorario)
                                            (asignatura-s ?asig)
                                            (profesor-s   ?prof)
                                            (aula-s       ?aul)
                                            (dia          ?ndia)
                                            (hora         ?nhora)
                                            (fecha        ?fecha) ))
                        )
                    )
                    ;(printout t "celdas dias " ?celdas-dia t)
                    ?celdas-dia
                )
            )
            (bind ?todas-celdas (create$))
            (for ?elem in$ ?celdas-horario do
                (bind ?todas-celdas (union$ ?todas-celdas ?elem))
            )
            ;(printout t ?todas-celdas t)
```

```
            ;(printout t "celdas horario " $?celdas-horario t)
            ; Nuevo objeto horario
            (bind ?IdAula (get-attribute-value
                                (nth$ (nth$ ?celdas (* ?naula ?*NumHoras*
?*NumDias*)) 3) ; (1ª celda).aula
                                numero))
            (bind ?nuevo-horario
                (assert (object =(string-to-symbol
                                    (string-append
                                        "hor-"
                                        (symbol-to-string ?aul)
                                        "_"
                                        (sprintf "curso%d" ?numcurso) ))
                            (instance-of horario)
                            (IdAula  ?IdAula) ; (1ª celda).aula.numero
                            (CeldaHorario-rel $?todas-celdas) )) )
        )
    )
    (bind ?nombre-curso (sprintf "cur-%d" ?numcurso))
    (assert (object =(string-to-symbol (sprintf "cur-%d" ?numcurso) )
            (instance-of ConjuntoHorarios)
            (nombre       ?nombre-curso)
            (grupo-m      $?grups)
            (horario-m    $?horarios-curso)
            (profesor-m   $?profs)
            (aula-m       $?auls)
            (asignatura-m $?asigs)
            (puntuacion   0) ))
    (retract ?f)
    (if (= ?*NumSoluciones* ?*MaxSoluciones*) then
        (enable-ruleset preferencias)
        (disable-ruleset avances)
        (disable-ruleset constraints)
    )
)
```

Constraints

```
; Una asignatura debe ser impartida sólo por los profesores indicados para
ello

(define-rule reglAula_1
    (declare
        (salience 200)
        (ruleset constraints))
    (not (vuelta-atras))
    (solucion $?inicio ?ele )
    (object ?obj
        (instance-of elementos)
        (elemento ?ele)
        (posicion ?b&:(= ?b (+ 1 (length$ ?inicio)))))
    (test (not (eq (nth$ ?ele 1) null)))
    (not (object ?rel
        (instance-of imparte)
        (profesor-s ?profe&:(eq ?profe (nth$ ?ele 1)))
        (asignatura-s ?asig&:(eq ?asig (nth$ ?ele 2)))))
=>
    (set-attribute-value ?obj eliminado 1)
    (assert (vuelta-atras))
```

```
)

; En un aula, solo se pueden impartir asignaturas en las que dicho aula se
requiera.

(define-rule reglAula_2
    (declare
        (salience 200)
        (ruleset constraints))
    (not (vuelta-atras))
    (solucion $?inicio ?ele)
    (object ?obj
        (instance-of elementos)
        (elemento ?ele)
        (posicion ?b&:(= ?b (+ 1 (length$ ?inicio)))))
    (test (not (eq (nth$ ?ele 1) null)))
    (not (object ?rel
        (instance-of requiere)
        (asignatura-s ?asig&:(eq ?asig (nth$ ?ele 2)))
        (aula-s ?aul&:(eq ?aul (nth$ ?ele 3))) ))
=>
    (set-attribute-value ?obj eliminado 1)
    (assert (vuelta-atras))
)

;Regla que controla que las asignaturas que tienen 2 horas, no aparezcan 3
veces

(define-rule controlahoras1
    (declare
        (salience 200)
        (ruleset constraints))
    (not (vuelta-atras))
    (solucion $?inicio ?ele $?medio ?ele $?medio2 ?ele)
    (object ?obj
        (instance-of elementos)
        (elemento ?ele) ;buscamos un elemento igual al último de la solución
actual
        (posicion ?b&:(= ?b (+ 3 (length$ ?medio) (length$ ?inicio) (length$
?medio2)))))
=>
    (set-attribute-value ?obj eliminado 1)
    (assert (vuelta-atras))
)

; En un aula, solo se pueden impartir asignaturas que sean de esa aula y no
de otra.

(define-rule regla4
        (declare
            (salience 200)
            (ruleset constraints))
        (not (vuelta-atras))
        (solucion $?inicio ?ele1 ?ele2)
        (object ?obj1
            (instance-of elementos)
            (elemento ?ele1)
            (posicion ?a&:(= ?a (+ 1 (length$ ?inicio)))))
        (object ?obj2
```

```
            (instance-of elementos)
            (elemento ?ele2)
            (posicion ?b&:(= ?b (+ 2 (length$ ?inicio)))))
        (test (not (eq (nth$ ?ele1 3) (nth$ ?ele2 3))))
        (test (= (/ (- ?a 1) (* ?*NumHoras* ?*NumDias*)) (/ (- ?b 1) (*
?*NumHoras* ?*NumDias*))))
=>
    (set-attribute-value ?obj2 eliminado 1)
    (assert (vuelta-atras))
)

; Un profesor no puede estar en dos aulas a la vez (en la misma franja
horaria o celda)

(define-rule regla5
        (declare
            (salience 200)
            (ruleset constraints))
        (not (vuelta-atras))
        (solucion $?inicio ?ele1 $?medio ?ele2)
        (object ?obj1
            (instance-of elementos)
            (elemento ?ele1)
            (posicion ?a&:(= ?a (+ 1 (length$ ?inicio)))))
        (object ?obj2
            (instance-of elementos)
            (elemento ?ele2)
            (posicion ?b&:(= ?b (+ 2 (length$ ?medio) (length$ ?inicio)))))
        (test (not (eq (nth$ ?ele1 1) null)))
        (test (= (mod (/ (- ?a 1) ?*NumHoras*) ?*NumDias*)    ; mismo día,
horario distinto
                 (mod (/ (- ?b 1) ?*NumHoras*) ?*NumDias*) ))
        (test (= (mod (- ?a 1) ?*NumHoras*)                   ; misma hora
                 (mod (- ?b 1) ?*NumHoras*) ))
        (test (eq (nth$ ?ele1 1) (nth$ ?ele2 1))) ; mismo profesor
=>
    (set-attribute-value ?obj2 eliminado 1)
    (assert (vuelta-atras))
)

; No se puede rellenar un horario de un aula que ya se rellenó

(define-rule regla6
        (declare
            (salience 200)
            (ruleset constraints))
        (not (vuelta-atras))
        (solucion $?inicio ?ele1 ?ele2)
        (object ?obj1
            (instance-of elementos)
            (elemento ?ele1)
            (posicion ?a&:(= ?a (+ 1 (length$ ?inicio)))))
        (object ?obj2
            (instance-of elementos)
            (elemento ?ele2)
            (posicion ?b&:(= ?b (+ 2 (length$ ?inicio)))))
        (test (not (eq (nth$ ?ele1 1) null)))
        (test (eq (nth$ ?ele1 3) (nth$ ?ele2 3))) ;aulas iguales
```

```
        (test (not (= (/ (- ?a 1) (* ?*NumHoras* ?*NumDias*)) (/ (- ?b 1) (*
?*NumHoras* ?*NumDias*))))) ; distintos horarios
=>
    (set-attribute-value ?obj2 eliminado 1)
    (assert (vuelta-atras))
)

(define-rule regla7
        (declare
            (salience 200)
            (ruleset constraints))
        (not (vuelta-atras))
        (solucion $?inicio ?ele1 ?ele2)
        (aulas-completas $?ac)
        (object ?obj1
            (instance-of elementos)
            (elemento ?ele1)
            (posicion ?a&:(= ?a (+ 1 (length$ ?inicio)))))
        (object ?obj2
            (instance-of elementos)
            (elemento ?ele2)
            (posicion ?b&:(= ?b (+ 2 (length$ ?inicio)))))
        (test (not (eq (nth$ ?ele1 1) null)))
        (test (not (eq (nth$ ?ele1 3) (nth$ ?ele2 3)))) ; aulas distintas
        (test (member$ (nth$ ?ele2 3) ?ac)) ; si la segunda ya se rellenó
        (test (not (= (/ (- ?a 1) (* ?*NumHoras* ?*NumDias*)) (/ (- ?b 1) (*
?*NumHoras* ?*NumDias*)))))
=>
    (set-attribute-value ?obj2 eliminado 1)
    (assert (vuelta-atras))
)

; Una vez este completado un horario, las posiciones de ambos elementos
corresponderan con horarios
; distintos, el anterior, del horario completado, y el siguiente, de un nuevo
horario. Por tanto
; se ha de marcar el aula del horario anterior como completada.

(define-rule regla8
        (declare
            (salience 201)
            (ruleset constraints))
        (not (vuelta-atras))
        (solucion $?inicio ?ele1 ?ele2)
        ?f<-(aulas-completas $?ac)
        (object ?obj1
            (instance-of elementos)
            (elemento ?ele1)
            (posicion ?a&:(= ?a (+ 1 (length$ ?inicio)))))
        (object ?obj2
            (instance-of elementos)
            (elemento ?ele2)
            (posicion ?b&:(= ?b (+ 2 (length$ ?inicio)))))
        (test (not (eq (nth$ ?ele1 1) null)))
        (test (not (member$ (nth$ ?ele1 3) ?ac)))
        (test (not (= (/ (- ?a 1) (* ?*NumHoras* ?*NumDias*)) (/ (- ?b 1) (*
?*NumHoras* ?*NumDias*)))))
=>
    (bind ?x (nth$ ?ele1 3))
```

```
    (assert (aulas-completas $?ac ?x))
    (retract ?f)
    (assert (vuelta-atras))
)
```

Preferencias

```
(define-global ?*w1* = 0,4  :reset)

(define-global ?*w2* = 0,35 :reset)

(define-global ?*w3* = 0,25 :reset)

(define-function puntuacion-len-mat (?curso)
    (bind ?puntos 0)
    (for ?horario in$ (get-attribute-value ?curso horario-m) do
        (for ?celda in$ (get-attribute-value ?horario CeldaHorario-rel) do
            (if (and (< (get-attribute-value ?celda hora) 3)
                                        (or (eq (get-attribute-value ?celda
asignatura-s) Lengua5A)
                                            (eq (get-attribute-value
?celda asignatura-s) Lengua5B)
                                            (eq (get-attribute-value
?celda asignatura-s) Lengua5C)
                                            (eq (get-attribute-value
?celda asignatura-s) Matematicas5A)
                                            (eq (get-attribute-value
?celda asignatura-s) Matematicas5B)
                                            (eq (get-attribute-value
?celda asignatura-s) Matematicas5C) ))
            then
                (bind ?puntos (+ ?puntos 0,1))
            )
        )
    )
    ?puntos
)

(define-function puntuacion-horas-seguidas (?curso)
    (bind ?puntos 0)
    (for ?horario in$ (get-attribute-value ?curso horario-m) do
        (for ?celda in$ (get-attribute-value ?horario CeldaHorario-rel) do
                    (for ?celda2 in$ (get-attribute-value ?horario
CeldaHorario-rel) do
                        (if (and (= (get-attribute-value ?celda hora)
(+(get-attribute-value ?celda2 hora) 1))
                                        (= (get-attribute-value ?celda dia)
(get-attribute-value ?celda2 dia))
                                        (or
                                            (and (eq (get-attribute-value
?celda asignatura-s) Lengua5A) (eq (get-attribute-value ?celda2 asignatura-s)
Lengua5A))
                                            (and (eq (get-attribute-value
?celda asignatura-s) Lengua5B) (eq (get-attribute-value ?celda2 asignatura-s)
Lengua5B))
```

```
                                                (and (eq (get-attribute-value
?celda asignatura-s) Lengua5C) (eq (get-attribute-value ?celda2 asignatura-s)
Lengua5C))
                                                (and (eq (get-attribute-value
?celda asignatura-s) Matematicas5A) (eq (get-attribute-value ?celda2
asignatura-s) Matematicas5A))
                                                 (and (eq (get-attribute-value
?celda asignatura-s) Matematicas5B) (eq (get-attribute-value ?celda2
asignatura-s) Matematicas5B))
                                                (and (eq (get-attribute-value
?celda asignatura-s) Matematicas5C) (eq (get-attribute-value ?celda2
asignatura-s) Matematicas5C))
                                            )
                                    )
                            then
                                    (bind ?puntos (+ ?puntos 0,1))
                                )
                        )
                )
        )
    ?puntos
)

(define-rule preferencias
    (declare (ruleset preferencias))
    (object ?curso
        (instance-of ConjuntoHorarios))
    (not (curso-puntuado ?curso))
=>
    (set-attribute-value ?curso puntuacion (+ (* (float ?*w1*) (puntuacion-
len-mat ?curso)) (* (float ?*w2*) (puntuacion-horas-seguidas ?curso)) ))
    (assert (curso-puntuado ?curso))
)

(define-rule fin-preferencias
    (declare
        (ruleset preferencias)
        (salience -100))
=>
    (printout t "Ordenando las soluciones encontradas." t)
    (assert (maximo curso-0 -1))
    (disable-ruleset preferencias)
    (enable-ruleset ordenacion))
```

Ordenación

```
(define-rule ordena
    (declare
        (ruleset ordenacion)
        (salience 900))
    ?f<-(maximo ?curs ?max)
    (object ?c
        (instance-of ConjuntoHorarios)
        (puntuacion ?p))
    (test (> ?p ?max)) ; la puntuación es la mayor de las examinadas
=>
    (retract ?f)
```

```
    (assert (maximo ?c ?p))
)

(define-function muestra-horarios (?curso)
    (for ?horario in$ (get-attribute-value ?curso horario-m) do
                (printout t "+-------------------------------" t)
        (printout t "| Horario del aula: ")
        (for ?celda in$ (get-attribute-value ?horario CeldaHorario-rel) do
            (if (eq ?celda (nth$ (get-attribute-value ?horario CeldaHorario-
rel) 1)) then
                (printout t (get-attribute-value ?celda aula-s) t)
                (printout t "+-------------------------------" t)
            )
            (printout t "  Dia/hora: " (get-attribute-value ?celda dia) "/"
(get-attribute-value ?celda hora) " Prof/Asig: " (get-attribute-value ?celda
profesor-s) "/" (get-attribute-value ?celda asignatura-s) t)
        )
    )
)

(define-rule buscar-curso
    (declare
        (ruleset ordenacion)
        (salience 800))
    ?f<-(maximo ?curs ?max)
    (object ?curs
        (instance-of ConjuntoHorarios)
        (horario-m ?h))
=>
    (printout t "El mejor horario encontrado es el siguiente:" t)
    (muestra-horarios ?curs)
    (retract ?f)
    (printout t "Ejecucion terminada" t)
    (disable-ruleset ordenacion)
)
```