Programación Declarativa: Lógica y Restricciones

Programación Lógica con Restricciones

Constraint Logic Programming (*CLP*)

Mari Carmen Suárez de Figueroa Baonza

mcsuarez@fi.upm.es

POLITÉCNICA

Introducción (I): Restricciones

- Se usan para representar problemas
  - X+Y=20
  - X $\wedge$ Y es cierto
  - El tercer campo de la estructura de datos es mayor que el segundo
  - El hombre de amarillo que no tiene los ojos verdes
  - El asesino no conoce ningún detective que no lleve ropa oscura
- La solución a un conjunto de restricciones es
  - una asignación que cumple con las restricciones iniciales
    - Asesino: López, ojos verdes, pistola Magnum
  - o, un conjunto de restricciones
    - El asesino es uno de los que habían conocido a la cabaretera
  - o, no hay solución
    - Muerte natural

Introducción (II): CLP

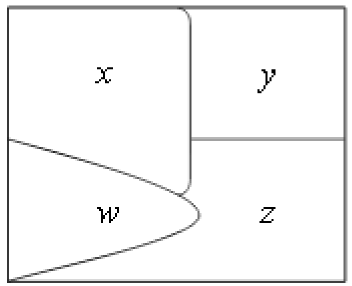
- **CLP (*Constraint Logic Programming*) (Programación Lógica con Restricciones)**: programación lógica más capacidad para manejar restricciones (que se resuelven por el sistema durante la ejecución)
  - Ofrece una forma declarativa para modelar problemas de satisfacción de restricciones
  - Lenguaje rico y potente para modelar problemas de optimización
- El modelado se basa en variables, dominios y restricciones
  - Dominios: reales, racionales, enteros, booleanos, estructuras, etc.
  - Tipos de expresiones en un dominio: +, *, $\wedge$, $\vee$, etc.
  - Tipos de restricciones permitidas para cada dominio
    - Ecuaciones, inecuaciones, etc. (=, ≤, ≥, <,>)
  - Algoritmos de resolución de restricciones: simplex, gauss, etc.

Introducción (III): CLP

- Un **problema de satisfacción de restricciones** se puede representar como un triple formado por
  - Un conjunto de variables V = {X1,...,Xn}
  - Un conjunto de posibles valores Di, que llamaremos dominio de Xi, para cada variable de V
  - Un conjunto de restricciones, normalmente binarias, Cij(Xi,Xj) que determinan los valores que las variables pueden tomar simultáneamente
- El objetivo es encontrar un valor para cada variable de manera que se satisfagan todas las restricciones del problema
  - Cada restricción limita el conjunto de asignaciones para las variables implicadas

Introducción (IV): Ejemplo 1

- **Colorear un mapa**: Sea un conjunto de colores posibles para colorear cada región del mapa, de manera que regiones adyacentes tengan distintos colores
  - Variables: {x; y; z; w}
    - una variable por cada región del mapa
  - Dominio: Rojo; Verde; Azul
    - conjunto de colores disponibles
  - Restricciones (definición intensional):
    - $x \neq y$, $x \neq w$, $x \neq z$, $y \neq z$, $w \neq z$
      - Para cada par de regiones contiguas existe una restricción sobre las variables correspondientes que no permite la asignación de valores idénticos a las variables
  - Una solución: (x,Rojo), (y,Verde), (w, Verde), (z, Azul)



Introducción (V): Ejemplo 2

- **El problema de las N-Reinas**: Se trata de colocar N reinas en un tablero de ajedrez de dimensión NxN, de forma que ninguna reina esté amenazada. Es decir, no puede haber dos reinas en la misma fila, misma columna, o misma diagonal
  - Variables: {xi}, i = 1..N
    - Cada columna es una variable y su valor representa la fila donde se coloca una reina
  - Dominio: {1,2,3, …, N} (para todas las variables)
    - Posibles filas donde colocar las reinas
  - Restricciones: $\forall$xi, xj, i $\neq$ j
    - xi $\neq$ xj (no en la misma fila)
    - xj – xi $\neq$ j – i (no en la misma diagonal)
    - xi – xj $\neq$ j – i (no en la misma diagonal)

Introducción (VI): CLP

- Ventajas
  - Mayor expresividad en el tratamiento de problemas
  - Diseño más uniforme y mayor efectividad
    - Puede ahorrar mucha codificación
  - Aumento de la eficiencia
    - Gracias a la reducción del espacio de búsqueda

      LP: generar-y-testear

      CLP: limitar-y-generar
- Desventajas
  - Algoritmos de resolución (simplex, gauss, etc.) complejos que pueden afectar al rendimiento
  - Necesidad de técnicas específicas para el tratamiento de los objetos

Sistemas de Restricciones: CLP(X)

- La semántica depende del dominio de las restricciones
- **CLP(X)**, donde X $\equiv(\Sigma,D,L,T)$
  - Signature Σ: conjunto de predicados y símbolos de función, junto con su aridad
  - $L\subseteq\Sigma$–formulae: restricciones
  - D: conjunto de elementos del dominio
  - Σ–structure D: proporciona el significado a los predicados y a los símbolos de función, y por tanto a las restricciones
  - T: teoría de primer orden (axiomatiza algunas propiedades de D)
  - (D,L): dominio de restricción
  - Ejemplo:
    - $\Sigma = \{0,1,\neg,\wedge,=\}$
    - D = {true, false}
    - (D,L) = BOOL (restricciones booleanas)

Restricciones en Ciao Prolog

- Planteadas como extensiones al sistema Prolog principal
  - Requieren la declaración inicial correspondiente
  - Restricciones sobre el dominio de los racionales
    - :- use_package(clpq).
  - Restricciones sobre el dominio de los reales
    - :- use_package(clpr).
  - Definen operadores especiales para expresar las restricciones
    - **.=.** , **.>.** , **.>=.,** etc.

CLP(X): Programas (I)

- Un **programa en CLP** es una colección de reglas de la forma
  - a ← b1, . . . , bn
    - donde a es un átomo y los bi's son átomos o restricciones
- Un hecho es una regla
  - a ← c
    - donde c es una restricción
- Un objetivo (o consulta) G es una conjunción de restricciones y átomos
- Cada restricción es una fórmula de primer orden construida con restricciones primitivas
  - p(t1,t2,…,tn), con términos t1, t2,…,tn y p símbolo de predicado, es una restricción primitiva

CLP(X): Programas (II)

- CLP puede utilizar la misma estrategia de ejecución que Prolog (primero en profundidad, y de izquierda a derecha) o una diferente
- La aritmética de Prolog (p.ej., is/2) puede permanecer o simplemente desaparecer, sustituida por la resolución de restricciones
- La sintaxis puede variar en los diferentes sistemas
  - Diferentes símbolos para las restricciones
    - *= para la unificación; #=, .=., etc., para las restricciones
    - Sobrecarga

      A = f (X, Y) se considera unificación

      A = X + Y se considera una restricción

CLP(ℜ): Un caso de estudio

- Prolog no puede resolver x-3 = y+5
- CLP(ℜ) es un lenguaje basado en Prolog, que incluye capacidades para resolver restricciones sobre números reales
  - Expresiones aritméticas lineales: números, variables y operadores (negación, suma, resta, multiplicación y división)
  - Ejemplo: t1 R t2, donde R = { >, ≥, =, ≤, <}
- CLP(ℜ) utiliza la misma estrategia de ejecución que Prolog
  - en profundidad y de izquierda a derecha
- CLP(ℜ) es capaz de resolver directamente (in)-ecuaciones lineales sobre números reales

Programación Lógica vs. CLP($\Re$) (I)

- Ejemplo: (Prolog)

  q(X, Y, Z) :- Z = f(X, Y).

  - ?- q(3, 4, Z).
    - Z = f(3,4)
  - ?- q(X, Y, f(3,4)).
    - X = 3, Y = 4
  - ?- q(X, Y, Z).
    - Z = f(X,Y)

- Ejemplo: (Prolog)

  p(X, Y, Z) :- Z is X + Y.

  - ?- p(3, 4, Z). **% modo in-in-out**
    - Z = 7
  - ?- p(X, 4, 7). **% modo out-in-in**
    - Instantiation Error

Programación Lógica vs. CLP (ℜ) (II)

- Ejemplo: (CLP(ℜ))

  p(X, Y, Z) :- Z .=. X + Y.

  - ?- p(3, 4, Z). **% modo in-in-out**
    - Z = 7

      yes
  - ?- p(X, 4, 7). **% modo out-in-in**
    - X = 3

      yes
  - ?- p(X, Y, 7). **% modo out-out-in**
    - X = 7 - Y

      yes

clp-Ejemplo.pl

Programación Lógica vs. CLP(ℜ) (III)

- Ejemplo: Reducción del espacio de búsqueda

  Prolog (generar y testear)

  - solution(X, Y, Z) :-

        p(X), p(Y), p(Z),
        test(X, Y, Z).

  - p(11). p(3). p(7). p(16). p(15). p(14).
  - test(X, Y, Z) :- Y is X + 1, Z is Y + 1.
  - *Consulta*: ?- solution(X, Y, Z).

        X = 14
        Y = 15
        Z = 16 ? ;
        no

  - 458 pasos (todas las soluciones: 465 pasos)

clp-ReduccionEspacioBusqueda.pl

Programación Lógica vs. CLP ($\Re$) (IV)

- Ejemplo: Reducción del espacio de búsqueda

CLP($\Re$) (generar y testear)

- solution(X, Y, Z) :-

  p(X), p(Y), p(Z),

  test(X, Y, Z).
- p(11). p(3). p(7). p(16). p(15). p(14).
- test(X, Y, Z) :- Y .=. X + 1, Z .=. Y + 1.
- Consulta: solution(X, Y, Z).

  Z = 16

  Y = 15

  X = 14 ?;

  no
- 458 pasos (todas las soluciones: 465 pasos)

Programación Lógica vs. CLP (ℜ) (V)

- <u>Ejemplo</u>: Reducción del espacio de búsqueda
  - Cambiar '*test(X, Y, Z)*' al principio (**restringir y generar**)
  - solution(X, Y, Z) :-
    test(X, Y, Z),
    p(X), p(Y), p(Z).
  - p(11). p(3). p(7). p(16). p(15). p(14).
  - *Prolog*: test(X, Y, Z) :- Y is X + 1, Z is Y + 1.
    - ?- solution(X, Y, Z).
      Instantiation Error
  - *CLP(ℜ)*: test(X, Y, Z) :- Y .=. X + 1, Z .=. Y + 1.
    - ?- solution(X, Y, Z).
      Z = 16
      Y = 15
      X = 14 ?;
      no
    - 6 pasos (todas las soluciones: 11 pasos)

Ejemplo: $E = \frac{1}{2} mv^2 + 9.81\ mh$

- En Prolog, un procedimiento que calcule cualquiera de las cuatro variables dadas las otras tres

```
energia(E,M,V,H):-
        number(M),
        number(V),
        number(H),
        E is 0.5*M*V*V + 9.81*M*H.

energia(E,M,V,H):-
        number(E),
        number(M),
        number(V),
        H is (E - 0.5*M*V*V)/(9.81*M).
```

```
energia(E,M,V,H):-
        number(E),
        number(M),
        number(H),
        V is sqrt((E - 9.81*M*H)/0.5*M).

energia(E,M,V,H):-
        number(E),
        number(V),
        number(H),
        M is E / (0.5*V*V + 9.81*H).
```

- En CLP(R)

```
% Ciao Prolog
:- use_package(clpr).

energia(E,M,V,H):-
        E .=. 0.5*M*V*V + 9.81*M*H.
```

```
% SWI Prolog
:- use_module(library(clpq)).

energia(E,M,V,H):-
        { E = 0.5*M*V*V + 9.81*M*H }.
```

Ecuaciones Lineales (CLP(ℜ))

- Vectores: listas de números
- Multiplicación de vectores (de números reales)
  - (x1, x2, . . . , xn) · (y1, y2, . . . , yn) = x1·y1 + … + xn·yn
  - prod([], [], 0).
    ```
    prod([X|Xs], [Y|Ys],P) :- P .=. X * Y + Rest,
                              prod(Xs, Ys, Rest).
    ```
- La unificación se convierte en resolución de restricciones
  - ?- prod([2, 3], [4, 5], K).
    - K = 23
  - ?- prod([2, 3], [5, X2], 22).
    - X2 = 4
  - ?- prod([2, 7, 3], [Vx, Vy, Vz], 0).
    - Vx = -1.5*Vz - 3.5*Vy
- Cualquier respuesta calculada es, en general, una ecuación sobre las variables de la consulta

Sistemas de Ecuaciones Lineales (CLP(ℜ))

- ¿Podemos resolver sistemas de ecuaciones?
  - 3x + y = 5
    x + 8y = 3
  - ?- prod([3, 1], [X, Y], 5), prod([1, 8], [X, Y], 3).
    - X = 1.6087, Y = 0.173913
- Se puede construir un predicado más general imitando la notación vectorial matemática A·x = b
  - system(_Vars, [], []).
  - system(Vars, [Co|Coefs], [Ind|Indeps]) :-
        prod(Vars, Co, Ind),
        system(Vars, Coefs, Indeps).
  - ?- system([X, Y], [[3, 1],[1, 8]],[5, 3]). **% podemos expresar y resolver sistemas de ecuaciones**
    - X = 1.6087, Y = 0.173913

clp-EcuacionesLineales.pl

CLP(ℜ): Ejemplo 1

- Una comida consiste en un entrante, un plato principal y un postre

  Suponemos que existe una base de datos con distintos tipos de comida y sus valores calóricos

  Se debe producir un menú con comida *light* (valor calórico menor de 10Kcal)

clp-lightMeal.pl

CLP($\Re$): Ejemplo 2

- Numeros de Fibonacci
  - F(0) = 0
  - F(1) = 1
  - F(n+2) = F(n+1) + F(n)
- Versión en Prolog
  - ```
    fib(0, 0).
    fib(1, 1).
    fib(N, F) :-
        N > 1,
        N1 is N - 1,
        N2 is N - 2,
        fib(N1, F1),
        fib(N2, F2),
        F is F1 + F2.
    ```
- Nota: Sólo se puede usar con el primer argumento instanciado (a un número)
    ```

CLP(ℜ): Ejemplo 2

- Versión CLP(ℜ) (sintácticamente similar a la anterior)
  - ```
    fib(0, 0).
    fib(1, 1).
    fib(N, F) :-
        N .>. 1,
        F1 .>=. 0,
        F2 .>=. 0,
        N1.=. N-1,
        N2.=. N-2,
        fib(N1, F1),
        fib(N2, F2),
        F.=.F1+F2.
    ```
- <u>Notas</u>:
  - Se incluyen todas las restricciones en el programa (F1 >= 0, F2 >= 0)
  - Se puede realizar la consulta “?- fib(N, F).”
    ```

Circuitos Analógicos RLC (CLP(ℜ))

- Análisis y síntesis de circuitos analógicos RLC
  - Circuito RLC: circuito lineal que contiene una resistencia eléctrica, una bobina (inductancia) y un condensador (capacitancia)
- Cada circuito se compone de
  - Un componente simple, o
  - Una conexión de circuitos más simples
  - Para simplificar, se suponen subredes conectadas solamente en paralelo y serie
- Se quiere relacionar la corriente (I), el voltaje (V) y la frecuencia (W) en estado estacionario
  - circuito(C,V,I,W) establece que a través de la red C, el voltaje es V, la corriente es I y la frecuencia es W
  - V e I deben modelarse como números complejos
    - la parte imaginaria tiene en cuenta la frecuencia angular

Circuitos Analógicos RLC (CLP(ℜ))

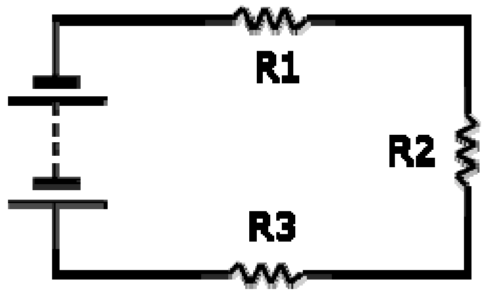
- Los números complejos X+Yi se modelan como c(X, Y)
- Operaciones básicas:
  - c_add (c (Re1, Im1), c (Re2, Im2), c (Re1 + Re2, Im1 + Im2)).
  - c_mult (c (Re1, Im1), c (Re2, Im2), c (Re3, Im3)):-

    Re3 .=. Re1 * Re2 - Im1 * Im2,

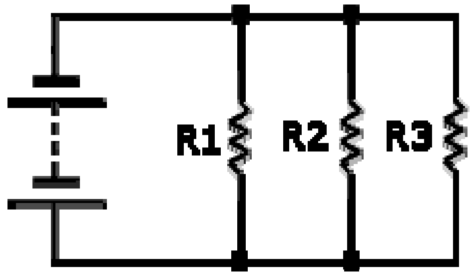
    Im3 .=. Re1 * Im2 + Re2 * Im1.
  - c_equal (c (R, I), c (R, I)).

Circuitos Analógicos RLC (CLP($\Re$))

- Circuito en serie:
  - circuit(serie(N1, N2), V, I, W) :-
    c_add(V1, V2, V),
    circuit(N1, V1, I, W),
    circuit(N2, V2, I, W).



- Circuito en paralelo:
  - circuit(parallel(N1, N2), V, I, W) :-
    c_add(I1, I2, I),
    circuit(N1, V, I1, W),
    circuit(N2, V, I2, W).



Circuitos Analógicos RLC (CLP(ℜ))

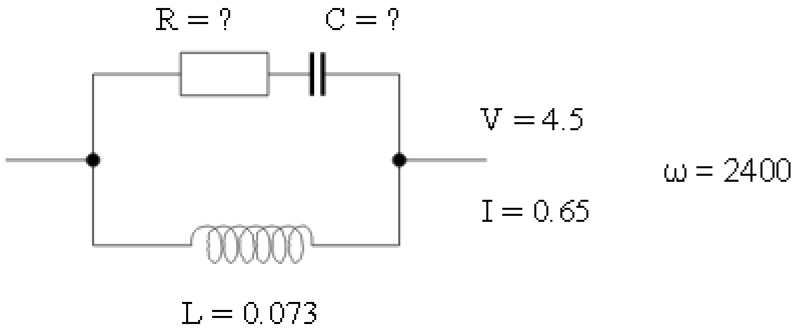
- Cada componente básico se modela como una unidad separada
  - Resistencia: V = I * (R+0i)
    - circuit (resistor(R), V, I, _W) :-

      c_mult(I,c(R,0), V).
  - Inductor: V = I * (0 + W·Li)
    - circuit (inductor(L), V, I, W) : -

      c_mult(I,c(0,W*L),V).
  - Condensador: V = I * (0 – (1/W·C)i)
    - circuit (capacitor(C),V,I,W) :-

      c_mult (I, c (0, -1 / (W * C)), V

Circuitos Analógicos RLC (CLP($\Re$)): Ejemplo



- ?- circuit(parallel(inductor(0.073),
  serie(capacitor(C), resistor(R))),
  c(4.5, 0), c(0.65, 0), 2400).
- R = 6.91229, C = 0.00152546

CLP(FD): Dominios Finitos

- Se asocia cada variable con un subconjunto finito de Z
  - Ejemplos: E $\in$ {-123,-10..4,10}
    - E :: [-123, -10..4, 10] (notación Eclipse)
    - E in {-123} $\vee$ (-10..4) $\vee$ {10} (notación SICStus)
    - E in [-123, -10..4, 10] (notación Ciao Prolog)
- Se puede
  - Establecer el dominio de una variable (in)
  - Realizar operaciones aritméticas (+,-,*,/) sobre las variables
  - Establecer relaciones lineales entre expresiones aritméticas (#=,#<,#=<)
  - Las operaciones y relaciones permiten reducir los dominios de las variables

CLP(FD): Dominios Finitos. Ejemplo

- ?- X #= A + B, A in 1..3, B in 3..7.

  X in 4..10, A in 1..3, B in 3..7
- No hay solución única
  - ?- X #= A - B, A in 1..3, B in 3..7.

    X in -6..0, A in 1..3, B in 3..7
    - El mínimo valor de X es el mínimo valor de A menos el máximo valor de B
    - Similar para el valor máximo de X
- Si incluimos más restricciones
  - ?- X #= A - B, A in 1..3, B in 3..7, X #>= 0.

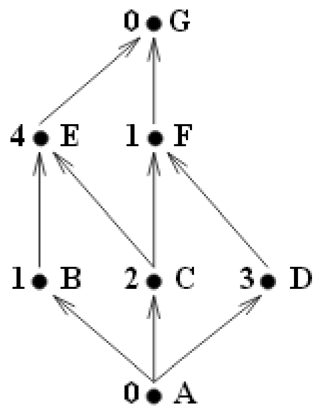
    A = 3, B = 3, X = 0

CLP(FD): Dominios Finitos

- Algunas primitivas útiles
  - fd_min(X,T): el término T es el valor mínimo en el dominio de la variable X
    - Se puede utilizar para reducir al mínimo una solución
    - ?- X # = A - B, A in 1..3, B in 3..7, fd_min (X, X).
      A = 1, B = 7, X = -6
  - domain(Variables,Min,Max): para englobar varias restricciones (in)
  - labeling(Options,Varlist)
    - Instancia variables en *VarList* con valores en sus dominios
    - *Options* indica el orden de búsqueda
- <u>Ejemplo</u>:
  - ? - X * Y * X + Y # = Z * Z, X #> = Y, domain ([X, Y, Z], 1,1000), labeling ([], [X, Y, Z]).
    X = 4, Y = 3, Z = 5
    X = 8, Y = 6, Z = 10
    X = 12, Y = 5, Z = 13
    ….

Ejemplo: Gestión de Proyectos

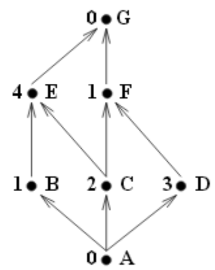
- El proyecto, cuyas dependencias y duraciones de tarea aparecen en la figura, debe finalizar en 10 unidades de tiempo (o menos)



- Restricciones:
  - pn1(A,B,C,D,E,F,G) :-
    A #>= 0, G #=< 10,
    B #>= A, C #>= A, D #>= A,
    E #>= B + 1, E #>= C + 2,
    F #>= C + 2, F #>= D + 3,
    G #>= E + 4, G #>= F + 1.
- Consulta:
  - ?- pn1(A,B,C,D,E,F,G).
    A in 0..4, B in 0..5, C in 0..4,
    D in 0..6, E in 2..6, F in 3..9, G in 6..10

Ejemplo: Gestión de Proyectos

- El proyecto, cuyas dependencias y duraciones de tarea aparecen en la figura, debe finalizar en 10 unidades de tiempo (o menos)



- Si queremos minimizar el tiempo total del proyecto

  ?- pn1(A,B,C,D,E,F,G), fd_min(G, G).

  - A = 0, B in 0..1, C = 0, D in 0..2,
    E = 2, F in 3..5, G = 6

Ejemplo: Pasatiempo

- Este pasatiempo consta de las siguientes afirmaciones:
  - Un alemán y un británico viven cada uno en una casa de diferente color y tienen diferentes mascotas
  - El alemán vive en la casa verde
  - Hay un perro en la casa blanca
- Se plantea la siguiente pregunta:
  - ¿Quién tiene un gato?

clp_pasatiempo.pl

Ejercicio: Pasatiempo

- Este pasatiempo consta de las siguientes afirmaciones
  - Un alemán, un británico y un sueco viven cada uno en una casa de color diferente, tienen diferentes mascotas y les gustan diferentes bebidas
  - El alemán vive en la casa verde
  - El sueco bebe café
  - Al británico no le gustan los gatos
  - Hay un perro en la casa blanca
  - El sueco no vive en la casa azul
  - Alguien bebe agua y tiene un pez
- Se plantea la siguiente pregunta
  - ¿Quién bebe té?

clp_pasatiempo_ejercicio2.pl

Ejercicio: SEND + MORE = MONEY

- … que quiere decir, en inglés, "Envía más dinero"
- Sustituye, en la suma siguiente, las letras por cifras (de 0 a 9) teniendo en cuenta que a cada letra distinta le corresponde una cifra diferente
  - Las variables *S*, *E*, *N*, *D*, *M*, *O*, *R*, *Y* representan dígitos entre 0 y 9
  - La tarea consiste en encontrar valores para estas variables de manera que la operación SEND+MORE=MONEY sea correcta
  - Todas las variables deben tomar valores únicos
  - Los números deben estar bien formados (lo que implica que *M*>0 y *S*>0)

$$\begin{array}{r} S\,E\,N\,D \\ +\,M\,O\,R\,E \\ \hline M\,O\,N\,E\,Y \end{array}$$

puzzle.pl

Programación Declarativa: Lógica y Restricciones

Programación Lógica con Restricciones

Constraint Logic Programming (*CLP*)

Mari Carmen Suárez de Figueroa Baonza

mcsuarez@fi.upm.es

POLITÉCNICA