Programación Declarativa:
Lógica y Restricciones

Programación Lógica con Restricciones

Constraint Logic Programming (*CLP*)

Mari Carmen Suárez de Figueroa Baonza
mcsuarez@fi.upm.es

POLITÉCNICA

ucción (I): Restricciones

an para representar problemas:

=20

Y es cierto

ercer campo de la estructura de datos es mayor que el segundo

ombre de amarillo que no tiene los ojos verdes

sesino no conoce ningún detective que no lleve ropa oscura

lución a un conjunto de restricciones es

asignación que cumple con las restricciones iniciales

sesino: López, ojos verdes, pistola Magnum

n conjunto de restricciones

l asesino es uno de los que habían conocido a la cabaretera

o hay solución

Muerte natural

ucción (II): CLP

(*Constraint Logic Programming*) (**Programación a con Restricciones)**: programación lógica más cidad para manejar restricciones (que se resuelven l sistema durante la ejecución)

ece una forma declarativa para modelar problemas de sfacción de restricciones

guaje rico y potente para modelar problemas de optimización

nodelado se basa en variables, dominios y cciones

ninios: reales, racionales, enteros, booleanos, estructuras, etc.

os de expresiones en un dominio: +, *, ∧, ∨

os de restricciones permitidas para cada dominio:

cuaciones, inecuaciones, etc. (=, ≤, ≥, <,>)

oritmos de resolución de restricciones: simplex, gauss, etc.

ucción (III): CLP

roblema de satisfacción de restricciones se puede
esentar como un triple formado por:

conjunto de variables V = {X1,...,Xn}

conjunto de posibles valores Di, que llamaremos dominio
Xi, para cada variable de V

conjunto de restricciones, normalmente binarias, Cij(Xi,Xj)
e determinan los valores que las variables pueden tomar
ultáneamente

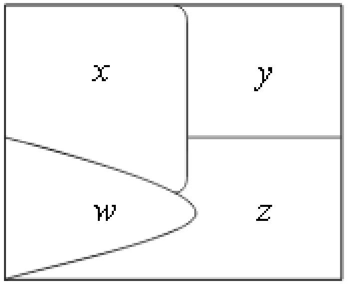
jetivo es encontrar un valor para cada variable de
era que se satisfagan todas las restricciones del
ema

a restricción limita el conjunto de asignaciones para las
ables implicadas

ucción (IV): Ejemplo 1

rear un mapa: Sea un conjunto de colores posibles colorear cada región del mapa, de manera que nes adyacentes tengan distintos colores

- iables: {x; y; z; w}
  - na variable por cada región del mapa
- ninio: Rojo; Verde; Azul
  - onjunto de colores disponible
- tricciones (definición intensional):
  - $\neq$ y, x $\neq$ w, x $\neq$ z, y $\neq$ z, w $\neq$ z
  - ara cada par de regiones contiguas existe una restricción sobre las ariables correspondientes que no permite la asignación de idénticos alores a las variables



solución: (x,Rojo), (y,Verde), (w, Verde), (z, Azul)

ucción (V): Ejemplo 2

oblema de las N-Reinas: Se trata de colocar N reinas n tablero de ajedrez de dimensión NxN, de forma ninguna reina esté amenazada. De esta forma no e haber dos reinas en la misma fila, misma nna, o misma diagonal

iables: {xi}, i = 1..N

ada columna es una variable y su valor representa la fila donde se coloca na reina

ninio: {1,2,3, …, N} (para todas las variables)

osibles filas donde colocar las reinas

tricciones: $\forall$xi, xj, i $\neq$ j

$\neq$ xj (no en la misma fila)

– xi $\neq$ j – i (no en la misma diagonal)

– xj $\neq$ j – i (no en la misma diagonal)

ucción (VI): CLP

ajas:

yor expresividad en el tratamiento de problemas

eño mas uniforme y mayor efectividad

uede ahorrar mucha codificación

nento de la eficiencia

racias a la reducción del espacio de búsqueda

LP: generar-y-testear

CLP: limitar-y-generar

entajas:

oritmos de resolución (simplex, gauss, etc.) complejos que de afectar al rendimiento

esidad de técnicas específicas para el tratamiento de los objetos

nas de Restricciones: CLP(X)

mántica depende del dominio de las restricciones: **()**, donde X ≡(Σ,D,L,T)

- nature Σ: conjunto de predicados y símbolos de función, junto su aridad
- –formulae: restricciones
- onjunto de elementos del dominio
- tructure D: proporciona el significado a los predicados y a los bolos de función, y por tanto a las restricciones
- eoría de primer orden (axiomatiza algunas propiedades de D)
-): dominio de restricción
- mplo:
  - = {0,1,¬,∧,=}
  - = {true, false}
  -),L) = BOOL (restricciones booleanas)

cciones en Ciao Prolog

eadas como extensiones al sistema Prolog principal

juieren la declaración inicial correspondiente

inen operadores especiales para expresar las restricciones

=. , .>. , .>=., etc.

tricciones sobre el dominio de los racionales

use_package(clpq).

tricciones sobre el dominio de los reales

use_package(clpr).

X): Programas (I)

rograma en CLP es una colección de reglas de la
a

b1, . . . , bn

onde a es un átomo y los bi’s son átomos o restricciones

echo es una regla

c

onde c es una restricción

objetivo (o consulta) G es una conjunción de
cciones y átomos

restricción es una fórmula de primer orden
truida con restricciones primitivas

,t2,…,tn), con términos t1, t2,…,tn y p símbolo de predicado, es
restricción primitiva

X): Programas (II)

- puede utilizar la misma estrategia de ejecución que g (primero en profundidad, y de izquierda a cha) o una diferente
- itmética de Prolog (p.ej., is/2) puede permanecer o emente desaparecer, sustituida por la resolución de cciones
- ntaxis puede variar en los diferentes sistemas:
  - erentes símbolos para las restricciones
    - = para la unificación; #=, .=., etc., para las restricciones
  - obrecarga:
    - = f (X, Y) se considera unificación
    - = X + Y se considera una restricción

ℜ): Un caso de estudio

g no puede resolver x-3 = y+5

ℜ) es un lenguaje basado en Prolog, que incluye cidades para resolver restricciones sobre números s

resiones aritméticas lineales: números, variables y operadores gación, suma, resta, multiplicación y división)

mplo: t1 R t2, donde R = { >, ≥, =, ≤, <, =}

ℜ) utiliza la misma estrategia de ejecución que g

nero en profundidad, de izquierda a derecha

ℜ) es capaz de resolver directamente (in)-ciones lineales sobre números reales

mación Lógica vs. CLP($\Re$) (I)

plo: (Prolog)

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

(3, 4, Z).

= f(3,4)

(X, Y, f(3,4)).

= 3, Y = 4

(X, Y, Z).

= f(X,Y)

plo: (Prolog)

/, Z) :- Z is X + Y.

(3, 4, Z). **% modo in-in-out**

= 7

(X, 4, 7). **% modo out-in-in**

nstantiation Error

mación Lógica vs. CLP (ℜ) (II)

plo: (CLP(ℜ))

/, Z) :- Z .=. X + Y.

(3, 4, Z). **% modo in-in-out**

= 7

es

(X, 4, 7). **% modo out-in-in**

= 3

es

(X, Y, 7). **% modo out-out-in**

= 7 - Y

es

o.pl

mación Lógica vs. CLP(ℜ) (III)

plo: Reducción del espacio de búsqueda

g (generar y testear)

```
tion(X, Y, Z) :-
(X), p(Y), p(Z),
est(X, Y, Z).
1).        p(3).       p(7).       p(16).    p(15).    p(14).
(X, Y, Z) :- Y is X + 1, Z is Y + 1.
```

sulta: ?- solution(X, Y, Z).

= 14

= 15

= 16 ? ;

o

pasos (todas las soluciones: 465 pasos)

cionEspacioBusqueda.pl

…mación Lógica vs. CLP ($\Re$) (IV)

…plo: Reducción del espacio de búsqueda

…$\Re$) (generar y testear)

…ution(X, Y, Z) :-

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

test(X, Y, Z).

…1). p(3). p(7). p(16). p(15). p(14).

…(X, Y, Z) :- Y .=. X + 1, Z .=. Y + 1.

…sulta: solution(X, Y, Z).

… = 16

… = 15

… = 14 ?;

…o

… pasos (todas las soluciones: 465 pasos)

mación Lógica vs. CLP ($\Re$) (V)

<u>plo</u>: Reducción del espacio de búsqueda

nbiar '*test(X, Y, Z)*' al principio (**restringir y generar**):

```
ution(X, Y, Z) :-
            test(X, Y, Z),
            p(X), p(Y), p(Z).
1).         p(3).     p(7).     p(16).    p(15).    p(14).
```

log: test(X, Y, Z) :- Y is X + 1, Z is Y + 1.

- solution(X, Y, Z).

nstantiation Error

($\Re$): test(X, Y, Z) :- Y .=. X + 1, Z .=. Y + 1.

- solution(X, Y, Z).

= 16

= 15

= 14 ?;

o

pasos (todas las soluciones: 11 pasos)

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

rolog, un procedimiento que calcule cualquiera de
ıatro variables dadas las otras tres:

```
/,H):-
er(M),
er(V),
er(H),
 0.5*M*V*V + 9.81*M*H.

/,H):-
er(E),
er(M),
er(V),
 (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).
```

CLP(R):

```
g
ge(clpr).

/,H):-
. 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 }.
```

iones Lineales (CLP(ℜ))

ores: listas de números

iplicación de vectores (de números reales):

x2, . . . , xn) · (y1, y2, . . . , yn) = x1·y1 + ... + xn·yn

d([], [], 0).

d([X|Xs], [Y|Ys],P) :- P .=. X * Y + Rest,

prod(Xs, Ys, Rest).

ificación se convierte en resolución de restricciones

rod([2, 3], [4, 5], K).

= 23

rod([2, 3], [5, X2], 22).

2 = 4

rod([2, 7, 3], [Vx, Vy, Vz], 0).

x = -1.5*Vz - 3.5*Vy

quier respuesta calculada es, en general, una ción sobre las variables de la consulta

nas de Ecuaciones Lineales (CLP(ℜ))

emos resolver sistemas de ecuaciones?

- y = 5

8y = 3

```
rod([3, 1], [X, Y], 5), prod([1, 8], [X, Y], 3).
 = 1.6087, Y = 0.173913
```

uede construir un predicado más general imitando la
ción vectorial matemática A·x = b:

```
tem(_Vars, [], []).
tem(Vars, [Co|Coefs], [Ind|Indeps]) :-
rod(Vars, Co, Ind),
ystem(Vars, Coefs, Indeps).
ystem([X, Y], [[3, 1],[1, 8]],[5, 3]).   % podemos expresar y resolver sistemas de ecuaciones
 = 1.6087, Y = 0.173913
```

onesLineales.pl

R): Ejemplo

comida consiste en un entrante, un plato principal y
ostre

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

ebe producir un menú con comida *light* (valor
ico menor de 10Kcal)

eal.pl

tos Analógicos RLC (CLP(ℜ))

sis y sintesis de circuitos analógicos RLC

uito RLC: circuito lineal que contiene una resistencia eléctrica, bobina (inductancia) y un condensador (capacitancia)

circuito se compone de:

componente simple, o

conexión de circuitos más simples

a simplificar, se suponen subredes conectadas solamente en alelo y serie

uiere relacionar la corriente (I), el voltaje (V) y la encia (W) en estado estacionario

uito(C,V,I,W) establece que a través de la red C, el voltaje es V, orriente es I y la frecuencia es W

I deben modelarse como números complejos

parte imaginaria tiene en cuenta la frecuencia angular

tos Analógicos RLC (CLP(ℜ))

úmeros complejos X+Yi se modelan como c(X, Y)

aciones básicas:

dd (c (Re1, Im1), c (Re2, Im2), c (Re1 + Re2, Im1 + Im2)).

ıult (c (Re1, Im1), c (Re2, Im2), c (Re3, Im3)):-

e3 = Re1 * Re2 - Im1 * Im2,

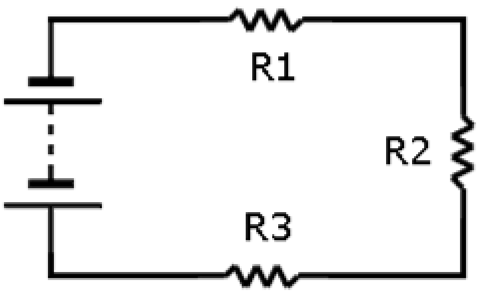
m3 = Re1 * Im2 + Re2 * Im1.

qual (c (R, I), c (R, I)).

tos Analógicos RLC (CLP(ℜ))

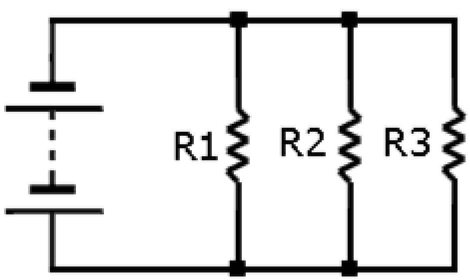
ito en serie:

```
uit(serie(N1, N2), V, I, W) :-
_add(V1, V2, V),
ircuit(N1, V1, I, W),
ircuit(N2, V2, I, W).
```



ito en paralelo:

```
uit(parallel(N1, N2), V, I, W) :-
_add(I1, I2, I),
ircuit(N1, V, I1, W),
ircuit(N2, V, I2, W).
```



tos Analógicos RLC (CLP(ℜ))

componente básico se modela como una unidad
rada:

istencia: V = I * (R+0i)

```
rcuit (resistor(R), V, I, _W) :-
            c_mult(I,c(R,0), V).
```

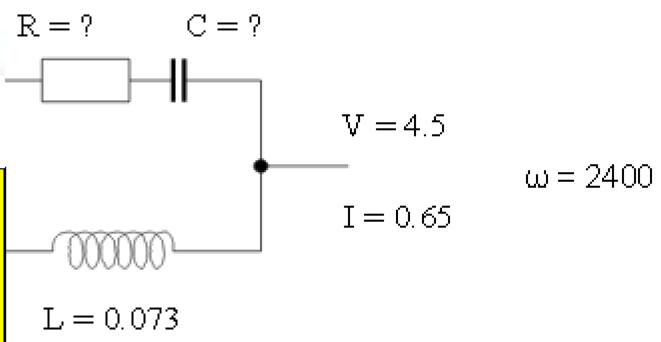
uctor: V = I * (0 + W·Li)

```
rcuit (inductor(L), V, I, W) : -
            c_mult(I,c(0,W*L),V).
```

densador: V = I * (0 – (1/W·C)i)

```
rcuit (capacitor(C),V,I,W) :-
            c_mult (I, c (0, -1 / (W * C)), V
```

tos Analógicos RLC (CLP(ℜ)): Ejemplo



cuit(parallel(inductor(0.073),
serie(capacitor(C), resistor(R))),
c(4.5, 0), c(0.65, 0), 2400).

.91229, C = 0.00152546

FD): Dominios Finitos

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

FD): Dominios Finitos. Ejemplo

#= A + B, A in 1..3, B in 3..7.

..10, A in 1..3, B in 3..7

ay solución única

#= A - B, A in 1..3, B in 3..7.

-6..0, A in 1..3, B in 3..7

l mínimo valor de X es el mínimo valor de A menos el máximo valor de B

imilar para el valor máximo de X

luimos más restricciones:

#= A - B, A in 1..3, B in 3..7, X #>= 0.

3, B = 3, X = 0

FD): Dominios Finitos

ıas primitivas útiles :

min(X,T): el término T es el valor mínimo en el dominio de la able X

e puede utilizar para reducir al mínimo una solución

- X # = A - B, A in 1..3, B in 3..7, fd_min (X, X).

= 1, B = 7, X = -6

ıain(Variables,Min,Max): para englobar varias restricciones (in)

eling(Options,Varlist):

ıstancia variables in VarList con valores en sus dominios

ptions indica el orden de búsqueda

plo:

X * Y * X + Y # = Z * Z, X #> = Y, domain ([X, Y, Z], 1,1000), labeling [X, Y, Z]).

4, Y = 3, Z = 5

8, Y = 6, Z = 10

12, Y = 5, Z = 13

plo: Gestión de Proyectos

oyecto, cuyas dependencias y duraciones de tarea ecen en la figura, debe finalizar en 10 unidades de po (o menos)

icciones:

(A,B,C,D,E,F,G) :-

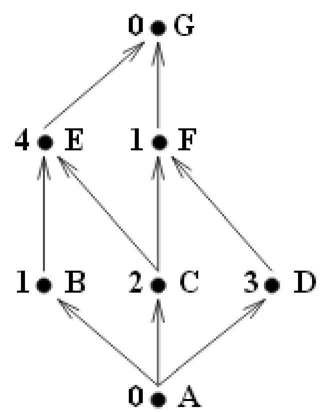
>= 0, G #=< 10,

>= A, C #>= A, D #>= A,

>= B + 1, E #>= C + 2,

>= C + 2, F #>= D + 3,

>= E + 4, G #>= F + 1.



ulta:

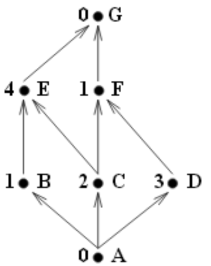
n1(A,B,C,D,E,F,G).

0..4, B in 0..5, C in 0..4,

0..6, E in 2..6, F in 3..9, G in 6..10

plo: Gestión de Proyectos

oyecto, cuyas dependencias y duraciones de tarea ecen en la figura, debe finalizar en 10 unidades de po (o menos)



eremos minimizar el tiempo total del proyecto:

1(A,B,C,D,E,F,G), fd_min(G, G).

0, B in 0..1, C = 0, D in 0..2,

2, F in 3..5, G = 6

cio: Pasatiempo

pasatiempo consta de las siguientes afirmaciones:

alemán, un británico y un sueco viven cada uno en una casa de or diferente, tienen diferentes mascotas y les gustan diferentes idas

lemán vive en la casa verde

ueco bebe café

ritánico no le gustan los gatos

un perro en la casa blanca

ueco no vive en la casa azul

uien bebe agua y tiene un pez

antea la siguiente pregunta:

ién bebe té?

empo_ejercicio.pl

cio: SEND + MORE = MONEY

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

```
   S E N D
 + M O R E
 ---------
 M O N E Y
```

Programación Declarativa:
Lógica y Restricciones

Programación Lógica con Restricciones

Constraint Logic Programming (*CLP*)

Mari Carmen Suárez de Figueroa Baonza
mcsuarez@fi.upm.es