Razonamiento con restricciones

Pedro Meseguer
IIIA – CSIC
Bellaterra

EVIA, La Coruña, 4 septiembre 2014

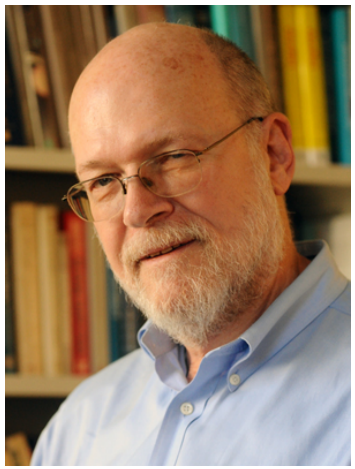
Sumario

- Un poco de historia:
  - comienzos
  - mejoras algorítmicas
  - extensiones
- *CSP*, ejemplos, aplicaciones
- Algoritmos de resolución: búsqueda / propagación / híbridos
- Mejoras: simetrías / restricciones globales
- Extensiones: restricciones contínuas / blandas / distribuidas
- Modelización & Programación con restricciones
- Temas actuales (*hot topics*):
  - sistemas multiagente, aprendizaje (*data mining*)
  - modelización, paralelismo
  - SAT, BDDs, MDDs, TDs, *catching,* tratabilidad
  - globales, *scheduling,* cuantificación, dinámismo, optimización multi

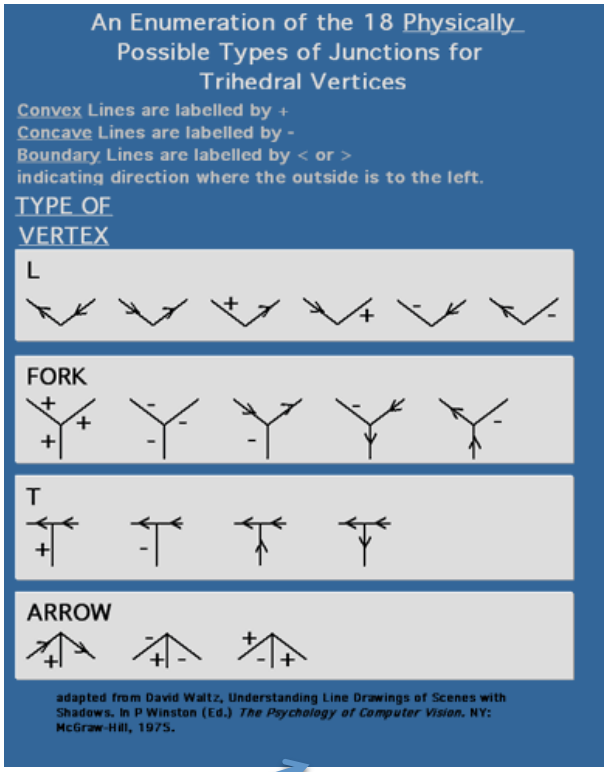
Los comienzos

MIT, comienzo de los años 70

un estudiante
de doctorado:
David Waltz



una tesis:
Generating Semantic Description from Drawings of Scenes with Shadows



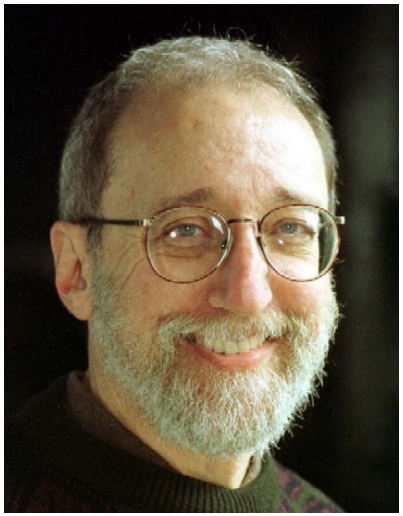
Descubren que los etiquetados de las imágenes reales han de satisfacer ciertas restricciones

Artículo en The Psycology of Computer Vision (P.H.Winston, 1975)

Formalización & Algoritmos

En los años 80 y 90, una enorme cantidad de trabajos sobre formalización y mejoras algoritmicas

Se fundamenta la idea de *constraint propagation*



Eugene Freuder
New Hampsire, USA
Cork, Irlanda

Alan Mackworth
British Columbia, Canadá

Rina Dechter
California, USA

También se desarrollan entornos de programación con restricciones: Ilog Solver, Chip,....

Extensiones & Aplicaciones

En los últimos 25 años:

- Extensiones: nuevos modelos de restricciones:
  - continuas
  - globales
  - blandas
  - distribuidas
- Conexiones con SAT / investigación operativa
- Aplicaciones específicas:
  - horarios (*timetabling*)
  - asignación temporal de recursos (*scheduling*)
  - optimización

Comunidad

- 1 asociación: ACP (http://www.a4cp.org/)
- 2 conferencias anuales: CP, CPAIOR
- 1 journal: Constraints
- Artículos de restricciones en:
  - IJCAI, AAAI, ECAI
  - Artificial Intelligence, JAIR

Definiciones

Red de restricciones (*CN*): (X, D, C)

- $X = \{x_1, x_2, \ldots, x_n\}$ variables
- $D = \{d_1, d_2, \ldots, d_n\}$ dominios (finitos)
- $C = \{c_1, c_2, \ldots, c_r\}$ restricciones

$c \in C$ $\quad var(c) = \{x_i, x_j, \ldots, x_k\}$ *ámbito*

$rel(c) \subseteq d_i \times d_j \times .. \times d_k$ *tuplas permitidas*

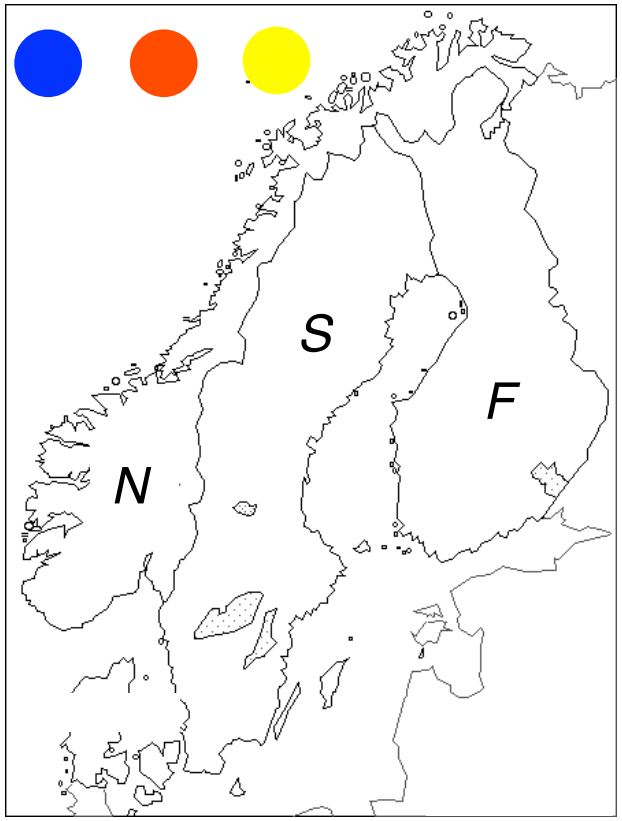
$arity(c) = |var(c)|$ (*unaria, binaria, ternaria,...*)

Problema de satisfacción de restricciones (*CSP*):

- Resolver la red (*CN*): asignar las variables satisfaciendo todas las restricciones: NP-completo

Ejemplo: Coloreado de mapas

Objetivo: Dado un mapa y un conjunto de colores, asignar un color a cada región tal que regiones adyacentes tengan colores diferentes

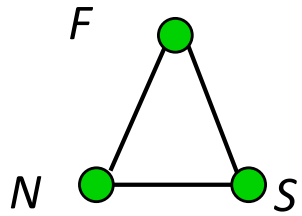


Formulación:

- Variables: regiones
- Dominios: colores
- Restricciones: if *adjacent*(x_i x_j)

  then $x_i \neq x_j$

Grafo de restricciones:



Ejemplo: n-reinas

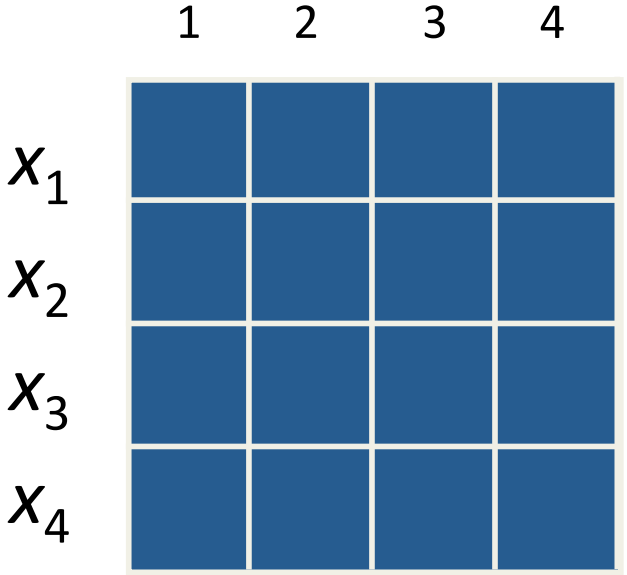
Objetivo: Colocar *n* reinas en un tablero de ajedrez *n* x *n* de forma que no se ataquen.

Formulación:

- Variables: una reina por fila
- Dominios: columnas
- Restricciones:

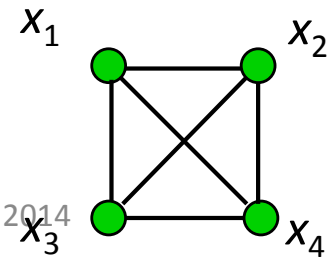
  diferentes columnas y diferentes diagonales

$x_i \neq x_j$ $\quad\quad$ $|\, x_i - x_j \,| \neq |\, i - j \,|$



4-reinas

Grafo de restricciones:



Importancia

CSP: modelo formal para expresar problemas

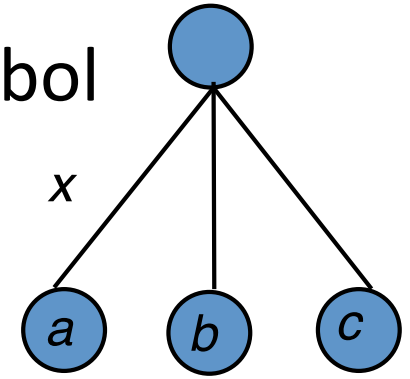
- *Artificial Intelligence*
  - *temporal reasoning*
- *Control Theory*
  - *controllers for sensory based robots*
- *Concurency*
  - *process comm. and synchr.*
- *Computer Graphics*
  - *geometric coherence*
- *Database Systems*
  - *constraint databases*
- *Bioinformatics*
  - *sequence alignment*
- *Operations research*
  - *optimization*

Real-life applications

- *Production planning*
- *Staff scheduling*
- *Resource allocation*
- *Circuit design*
- *Option trading*
- *DNA sequencing*
- *...*

Árbol de búsqueda

Espacio de estados: explorado como un árbol

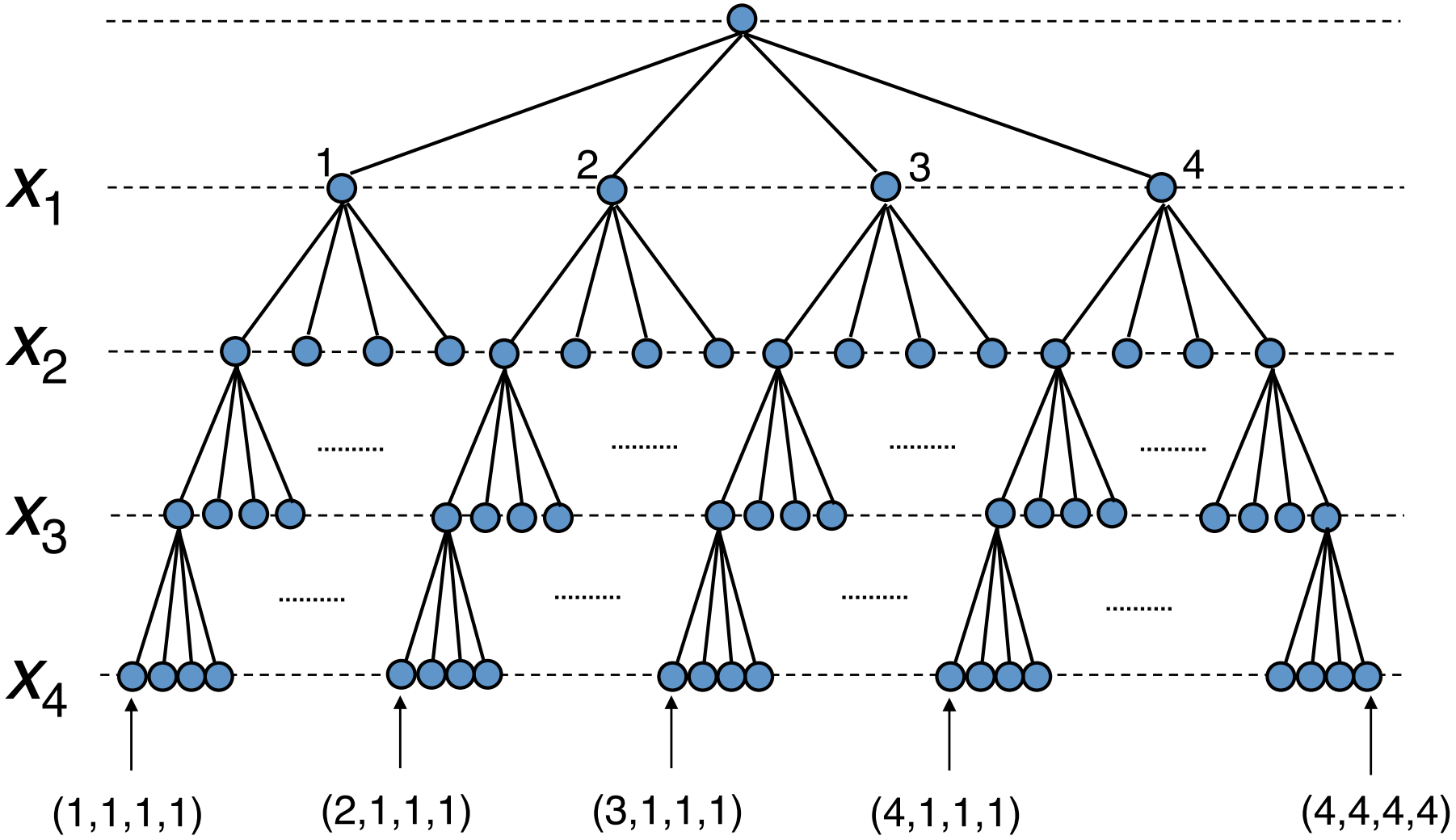


- raíz: vacío
- una variable por nivel
- sucesores de un nodo:
  - un sucesor por valor de la siguiente variable
  - significado: variable ← valor

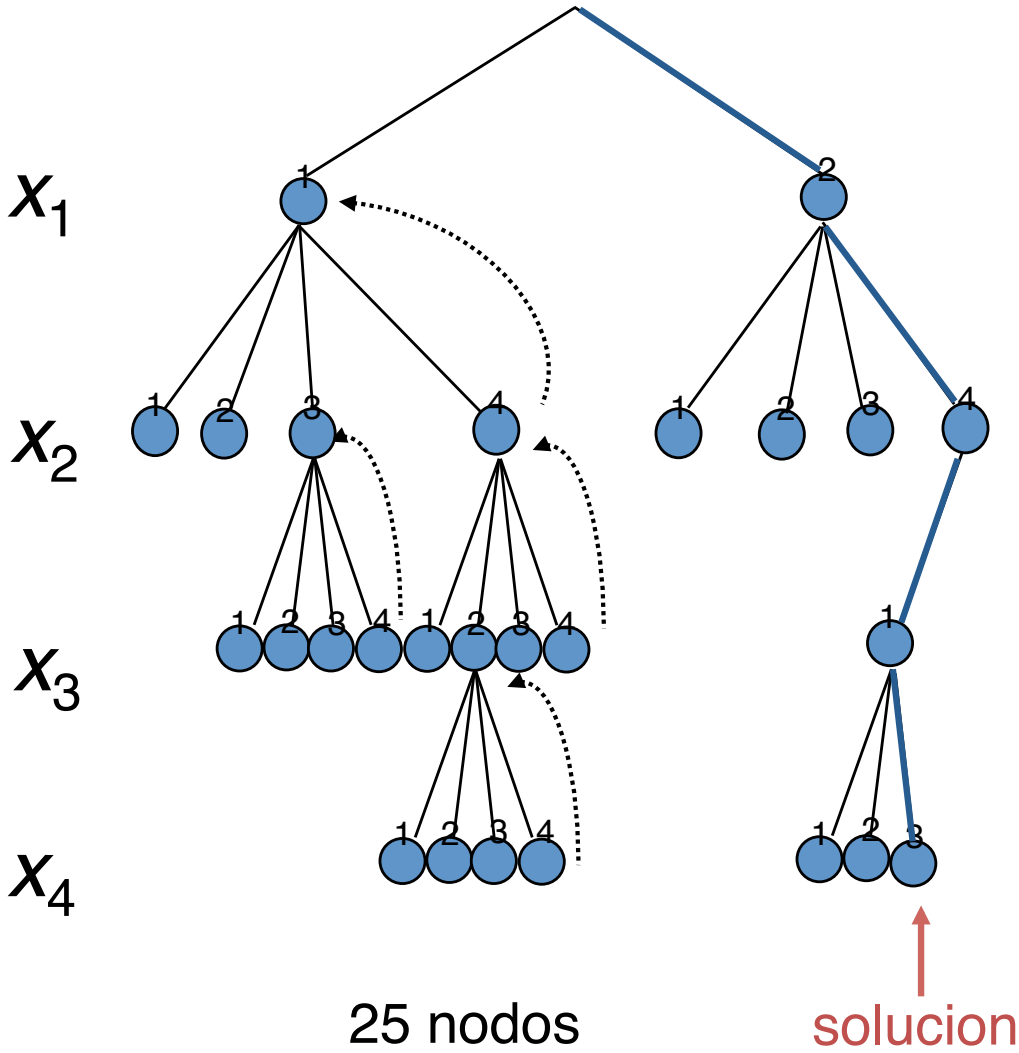
Árbol:

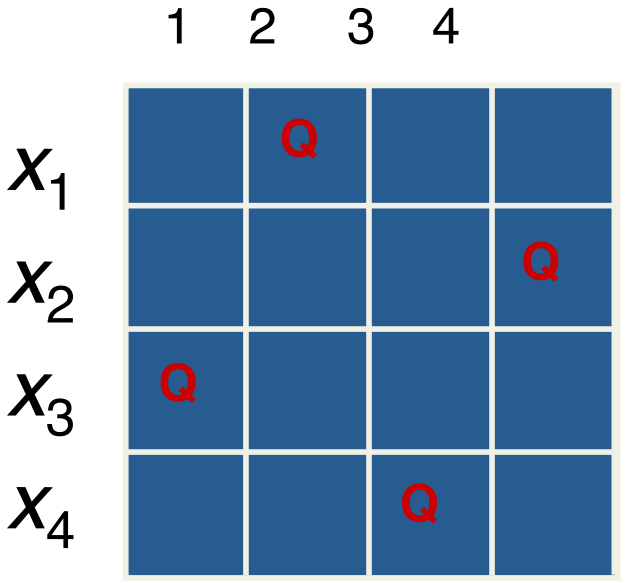
- cada rama define una asignación
- profundidad *n* (número de variables)
- factor de ramificación *d* (tamaño del dominio)

Árbol de búsqueda para 4-reinas



Backtracking sobre 4-reinas





Inferencia

operaciones legales sobre variables, dominios y restricciones

Inferencia: $P \longrightarrow P'$

- *P'* es equivalente a *P:* $Sol(P) = Sol(P')$
- *P'* es *presumiblemente más fácil* de resolver que *P*
  - espacio de estados más pequeño
  - restricciones más explícitas

Inferencia puede ser:

- completa: calcula la solución
  adaptive consistency
- incompleta: requiere algo de búsqueda
  arc consistency

Arco consistencia binaria (AC)

AC:

- Restricción C_{ij} es arco consistente direccional ($i \rightarrow j$)

  for all $a \in D_i$ *there is* $b \in D_j$ *such that* $(a, b) \in R_{ij}$

- Restricción C_{ij} es AC ssi es direccional AC en ambos sentidos

- Un problema es AC ssi cada restricción es AC

Filtrado por arco consistencia

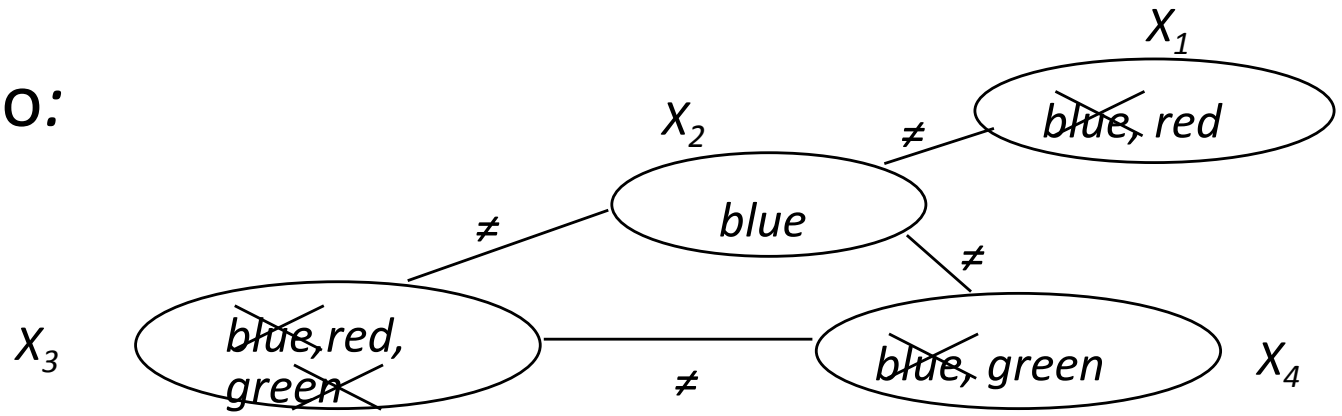
Si para cada $a \in D_i$ no existe $b \in D_j$ tal que $(a, b) \in R_{ij}$, a se puede eliminar de D_i

a no estará en ninguna solución

Filtrado de dominios (de Waltz):

- Eliminar valores arco inconsistentes
- Hasta que no haya cambios

Ejemplo:



Ejemplo: Ecuaciones sobre enteros

$x + y = 9$

$2x + 4y = 24$

x	0	1	2	3	4	5	**6**	7	8	9
y	0	1	2	**3**	4	5	6	7	8	9

punto fijo

propagación de restricciones

Híbridos: Búsqueda + AC

Idea:

- **Búsqueda**: *backtracking* (puede ser no-cronológico)
- **Inferencia**: cada nodo **AC** sobre algunas restricciones
  - **AC** descubre nogoods de tamaño 1
  - Valores no AC se eliminan

Efecto:

- Se reducen los dominios futuros: *menos nodos* a explorar
- **AC** en cada nodo: *más trabajo por* nodo
- Muy beneficioso: reduce *thrashing*

Mantener arco consistencia

MAC: combinación de

- Búsqueda: backtracking
- Inferencia: en cada nodo, **AC** sobre toda restricción
- Preproceso: subproblemas son **AC**

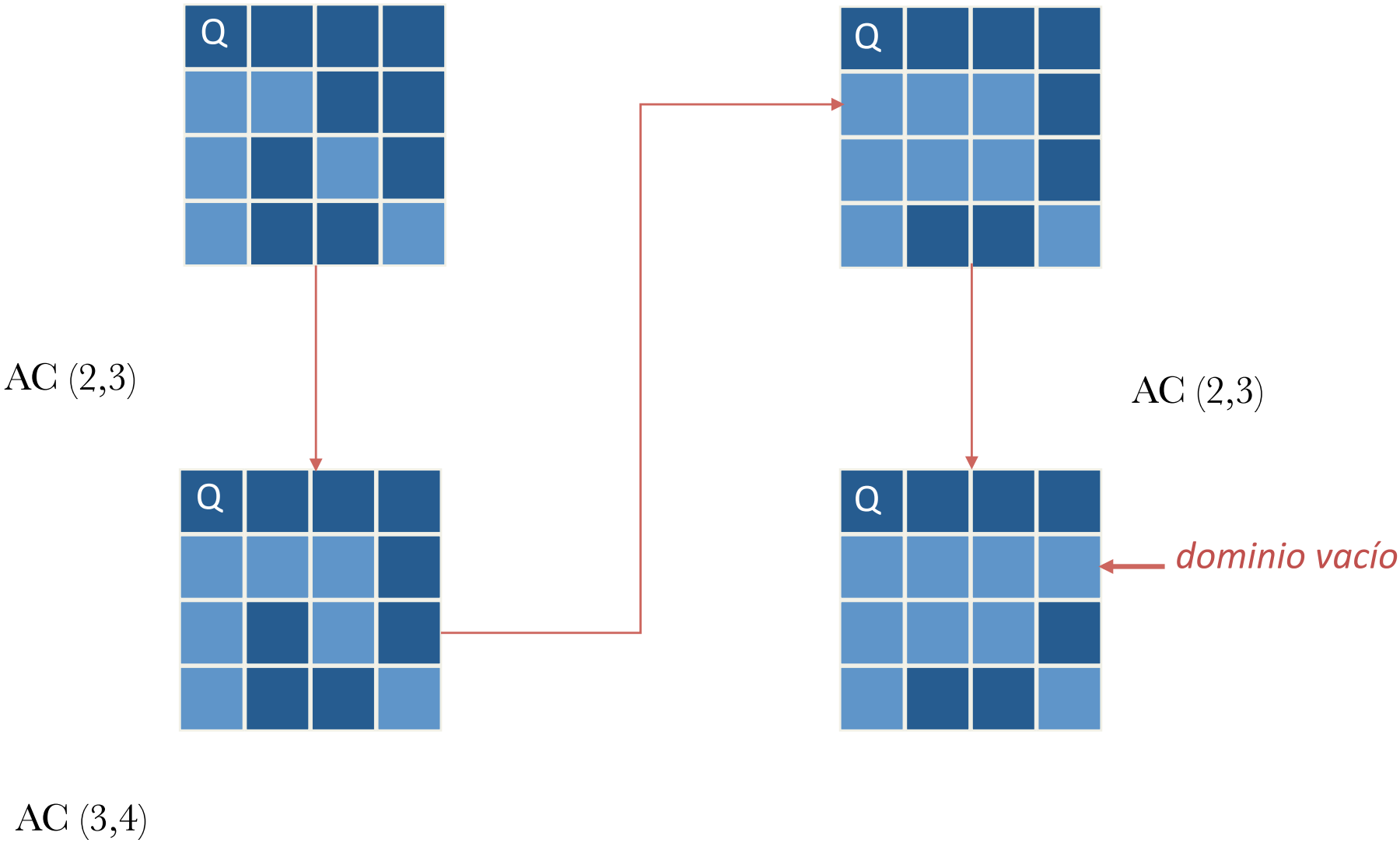
Cuando un dominio se vuelve vacío:

- No hay soluciones en la rama actual
- Poda y backtrack

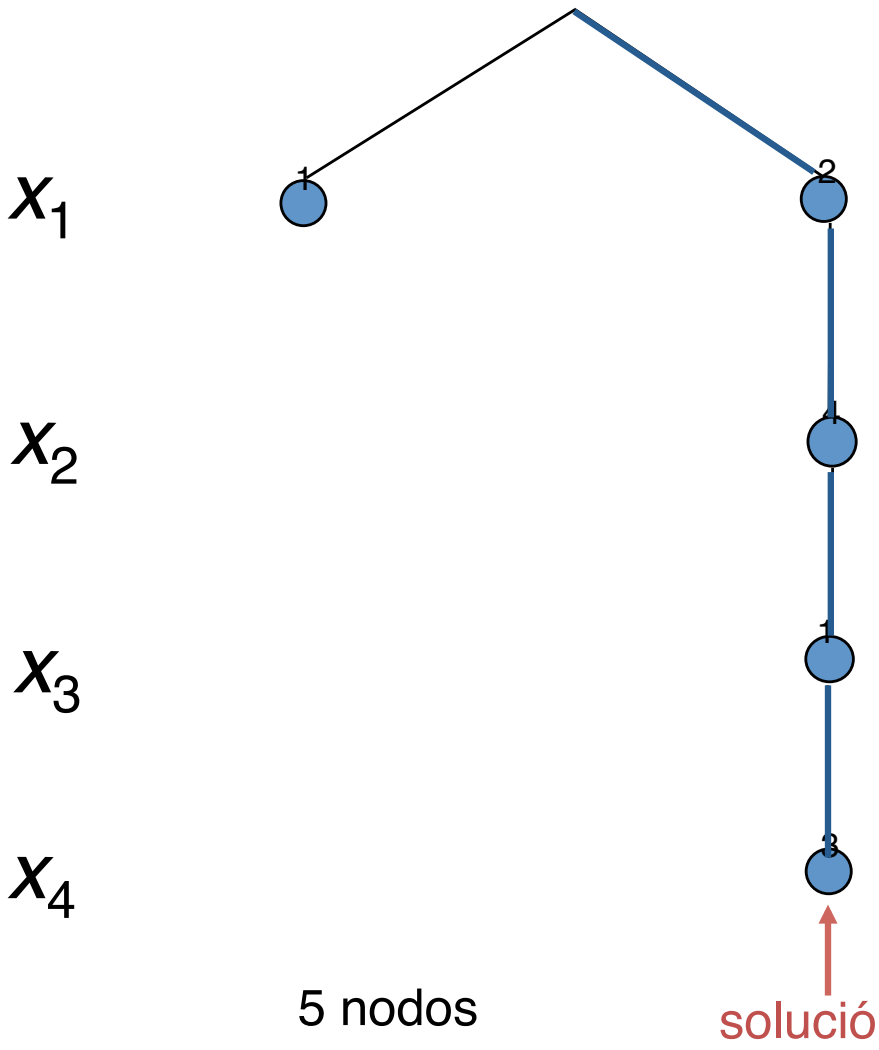
Precaución:

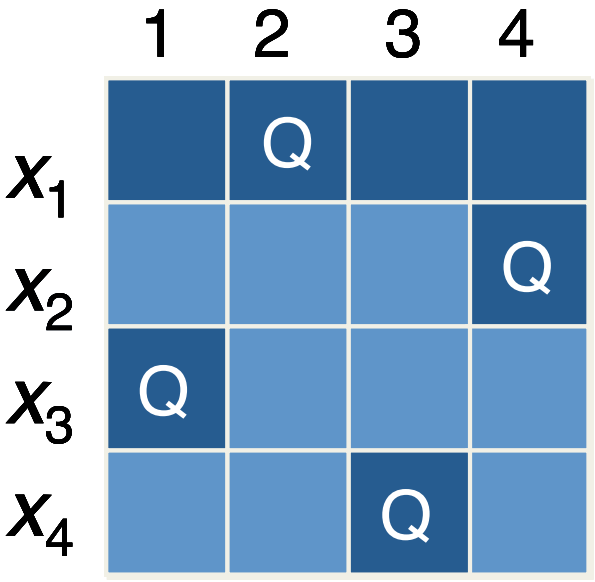
- Valores eliminados por **AC** en el nivel *i*, han de ser restaurados cuando se hace backtracking al nivel ≤ *i*

MAC vs FC: AC sobre futuras



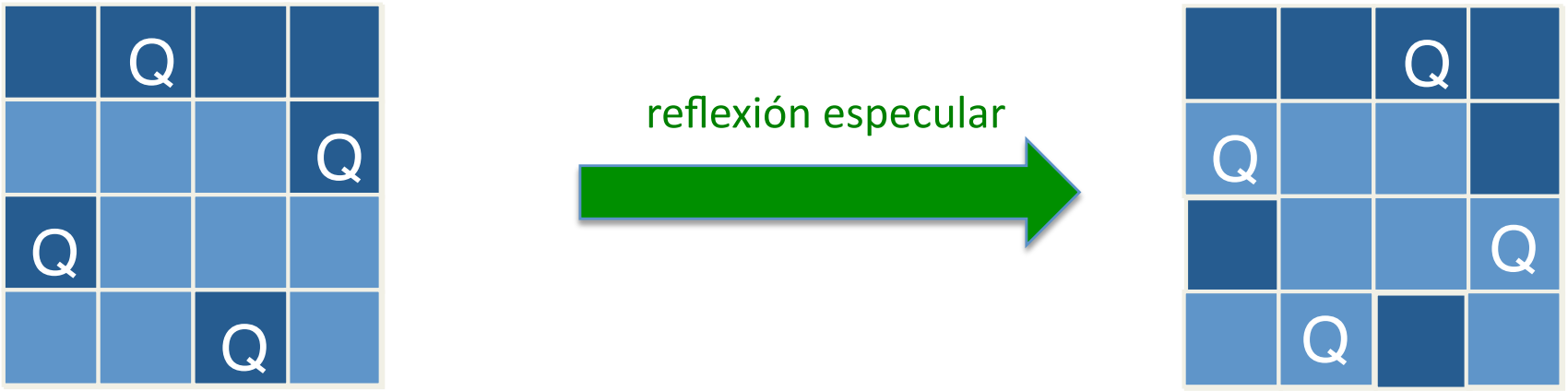
Ejemplo: MAC sobre 4-reinas





Simetrías

Simetría: operación que involucra a variables y valores, de forma que va de solución a solución



Dos tipos:

- *Solution symmetry:* mantiene el conjunto de soluciones
- *Problem symmetry:* mantiene el conjunto *C*

Explotación de simetrías

Estados simétricos en el espacio de estados:

- no hay que buscarlos todos: basta con encontrar uno
- sol → sol / no sol → no sol
- se simplifica mucho la búsqueda

Restricciones de ruptura de simetría entre variables:

- restricciones extra: ordenación lexicográfica de los valores que toman las variables;
  si x_1, x_2, x_3 variables simétricas
  $$x_1 > x_2 > x_3$$
- en general, no rompen todas las simetrías

Restricciones globales

Restricciones en la vida real: complejas, no-binarias

c es global ssi:

- aridad(*c*) > 2
- *c* es logicamente equivalente a $\{c_1, c_2, \ldots, c_k\}$ binarias
- **AC**(*c*) *poda más* que **AC**($c_1, c_2, \ldots, c_k$)

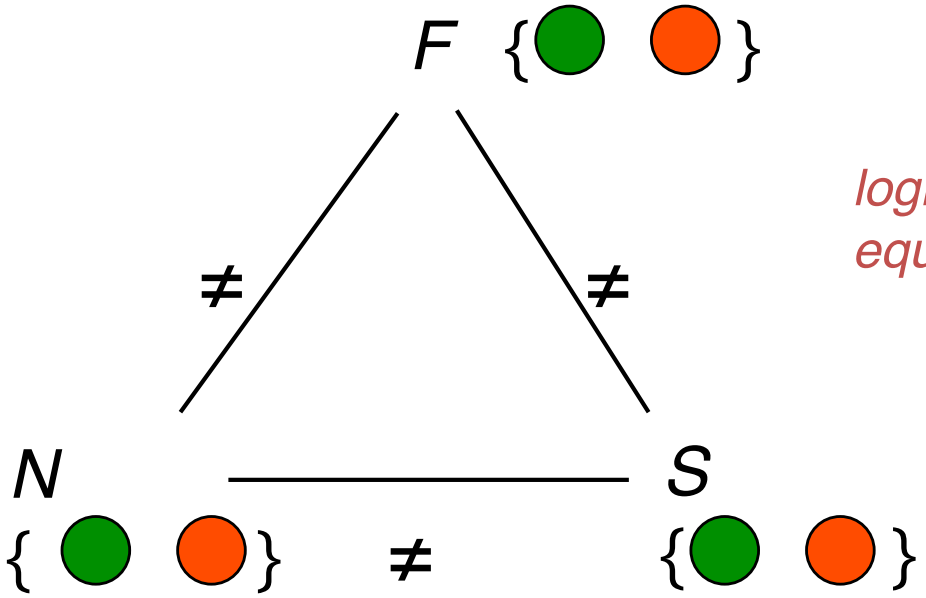
disminuye complejidad AC

Propagación:

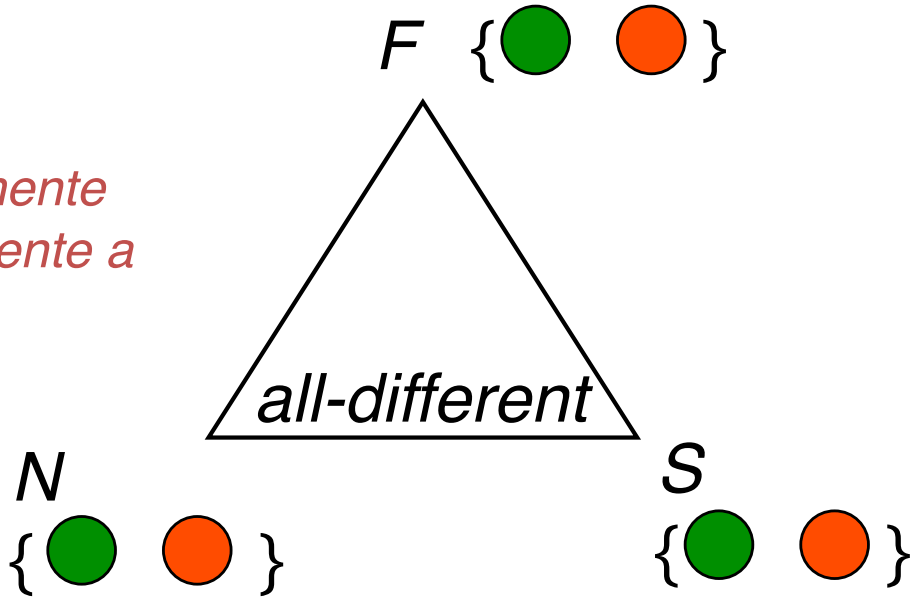
- algoritmos especializados
- explotan la semántica de la restricción

Ejemplo: *all-different*

Var: *F, N, S*; Val: {  }; Ctrs: $N \neq S \neq F \neq N$



logicamente equivalente a



3 restricciones binarias, son AC, no hay poda

1 restricción ternaria, no es AC, AC poda → dominio vacío
no hay solución!!

Restricciones contínuas

- Dominios contínuos infinitos: intervalos reales
- Restricciones: expresiones matemáticas
- Formalizan problemas de robótica
- Ejemplo:

$(x_1, x_2, x_3) \in R^3$

$x_1 + x_2 + x_3 = 0$

$x_1x_2 + x_1x_3 + x_2x_3 = 0$

$x_1x_2x_3 = 1$

Restricciones blandas

- Restricciones:
  - duras: prohiben totalmente una tupla de valores
  - blandas:
    - asignan un coste a cada tupla
    - objetivo: asignación global con el mínimo coste
- Problemas de optimización
- Algoritmo de *depth-first branch-and-bound*
- Cotas:
  - esencial para poda
  - arco consistencia blanda -> buenas cotas

Restricciones distribuidas

- Variables/restricciones, distribuidas entre distintos agentes
- No se pueden unir en un agente por:
  - confidencialidad
  - entorno muy dinámico
  - evitar un único punto de fallo

- Se encuentra una solución global vía paso de mensajes
  - satisfacción
  - optimización

Modelización

- De un problema real a un problema formalizado
- Impacto considerable en la eficiencia
- Determinar:
  - Variables, dominios
  - Restricciones
- Ejemplo: n-reinas
  - Modelo 1: cada casilla es un booleano
  - Modelo 2: cada reina en una fila, sin *all-diff*
  - Modelo 3: cada reina en una fila, con *all-diff*

Modelos de n-reinas

	Modelo 1	Modelo 2	Modelo 3
Search space $d^{\#vars}$	2^{n2} 4 65,536 12 1.27 E30 20 ERROR!!	n^n 256 1.00 E10 1.05 E26	n^n 256 1.00 E10 1.05 E26
Restricciones número poda	n filas n columnas 2(n-1) diagonals	n columnas 2(n-1) diagonales Igual al modelo 1	1 all-diff 2(n-1) diagonales Más que el modelo 2

Programación con restricciones

Decisiones: entre alternativas

- algoritmo de búsqueda
- consistencia local

} *entrelazadas*

- heuristicas: variable / valor

Ejemplo: Coloreado de mapas

- orden de variables estático o dinámico?

Resolución eficiente:

- tamaño inicial del espacio de búsqueda razonable
- reducciones drásticas del espacio de búsqueda durante la misma

CP: Programación declarativa

Programación declarativa:

- Variables
- Dominios
- Restricciones

y se pide al SOLVER encontrar una solución!!

El SOLVER ofrece:

- Implementación para variables / dominios / restricciones
- Algoritmo híbrido: backtracking + inferencia incompleta
- Restricciones globales + propagación AC optimizada
- Detección de dominio vacío
- Heurísticas

Programación lógica con restricciones

Programación lógica:

- *Depth-first search*
- Unificación: *sustituye iguales por iguales* clausulas/database

Caso especial de resolución de restricciones

reemplazado por

Solver de restricciones más general → *Constraint Logic Programming*

Solvers:

- Chip, Eclipse, Mozart, Sictus Prolog (y muchos otros)

CP Librerias

Librería para ser incluida en tu programa:

- Programación imperativa

Proporciona:

- Objetos especiales:
  - Variables / Dominios / restricciones (globales)
- Funciones especiales para encontrar:
  - Una solución / la siguiente solución

Librerías existentes:

- Ilog Solver, Choco

Sistemas multiagente

- *Distributed problem solving:* resolución de problemas entre múltiples agentes comunicados

- CSP distribuido:
  - variables en agentes
  - restricciones entre agentes
  - resolución distribuida: paso de mensajes
  - solución: optimización (mono, multi) objetivo

- Ejemplo: *Robocop rescue*

Data mining

- Tareas de *data mining*:
  - *frequent*
  - *closed*
  - *discriminative*
  - *cost-based*
- Se pueden expresar mediante restricciones
- Restricciones globales: papel central
- Programación con restricciones / eficiencia

Paralelismo

- Paralelismo en las CPUs actuales:
  - Procesadores especializados GPU
  - Varios cores por CPU

- Resolución de restricciones paralela:
  - no es inmediata
  - elemento secuencial: propagación entre restricciones
  - aprovechar el paralelismo en las CPUs

Gracias por vuestra atención!

Preguntas?