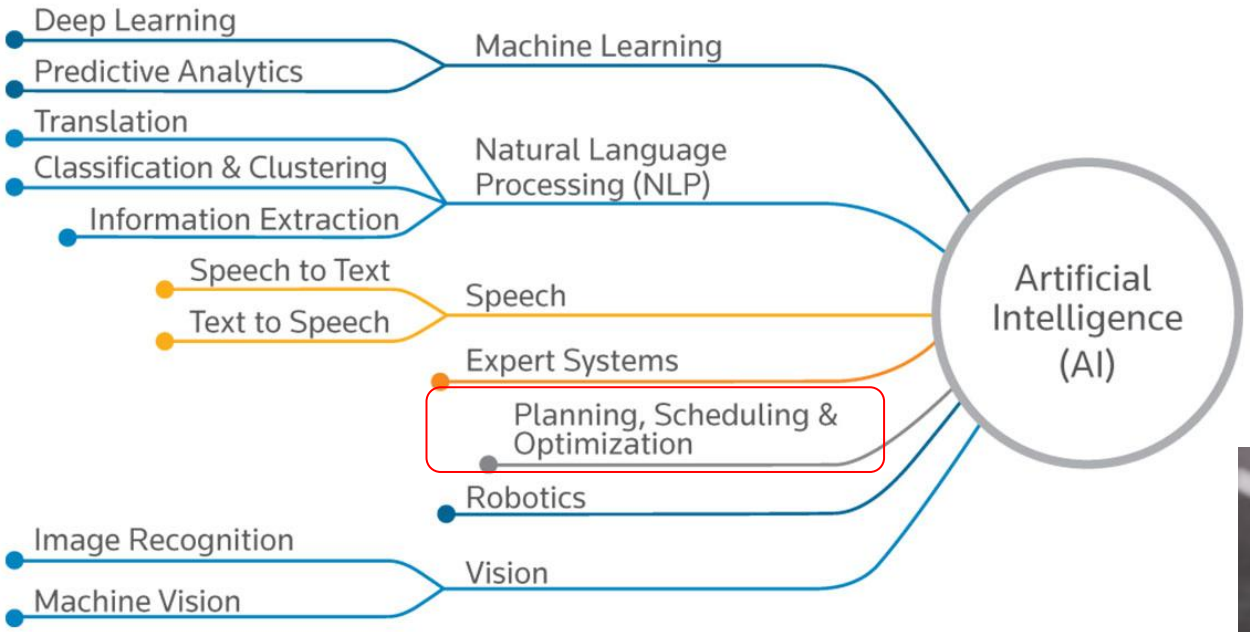
Puzzle solving con Intelligenza Artificiale, prima parte

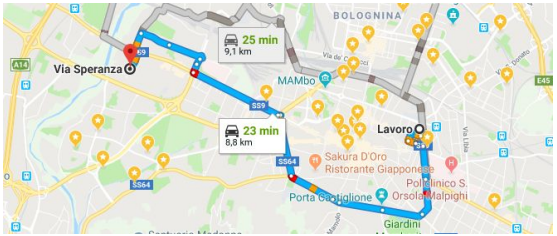
Dott. Tong Liu
Prof. Maurizio Gabbrielli
Alma Mater Studiorum - Università di Bologna

Contenuti Part 1

- Il Mondo dell'Intelligenza Artificiale
- Problema di soddisfacimento di vincolo
- Come istruire un computer ?
- Famiglie di linguaggi di programmazione
- Minizinc : un linguaggio di modellazione
- Elementi Minizinc
- Modellare i puzzle con Minizinc

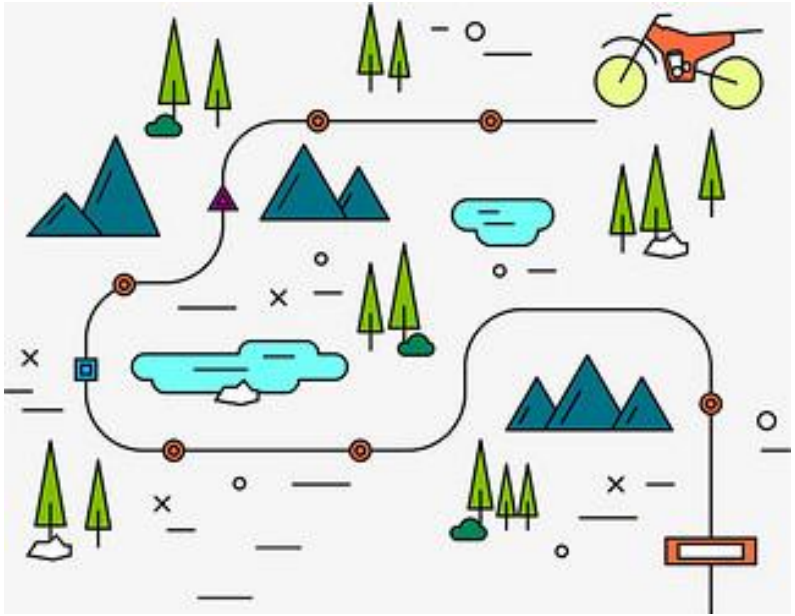
Il Mondo dell'Intelligenza Artificiale (AI)

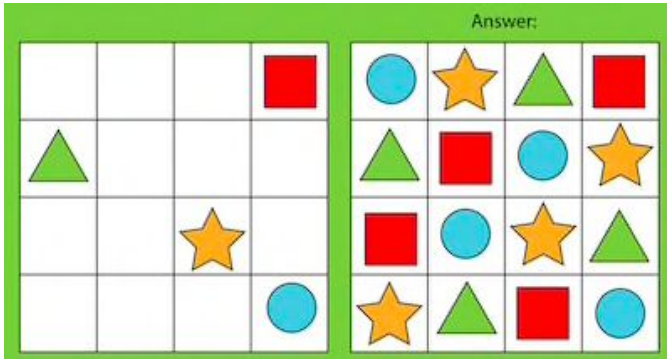


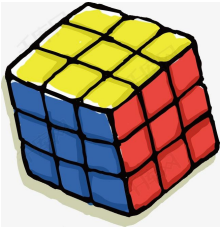


Constraint Satisfaction Problem

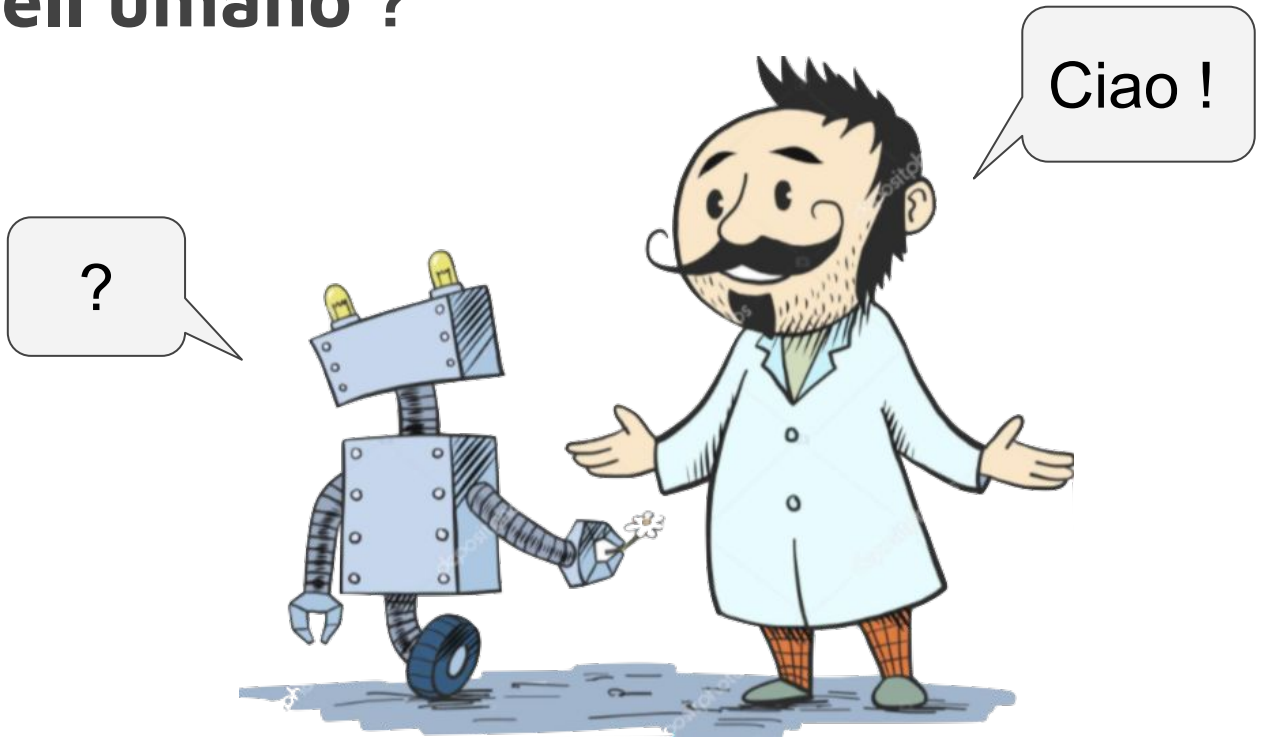
Problema di soddisfacimento di vincoli







Come il computer comprende i desideri dell'umano ?



Le modalità di interazioni

Ma può essere limitato ...

Codice di programmazione

Linguaggio Procedurale (Imperativo):

Dire al computer come fare

E.g. C, C++, Python

Linguaggio Dichiarativo (Descrittivo):

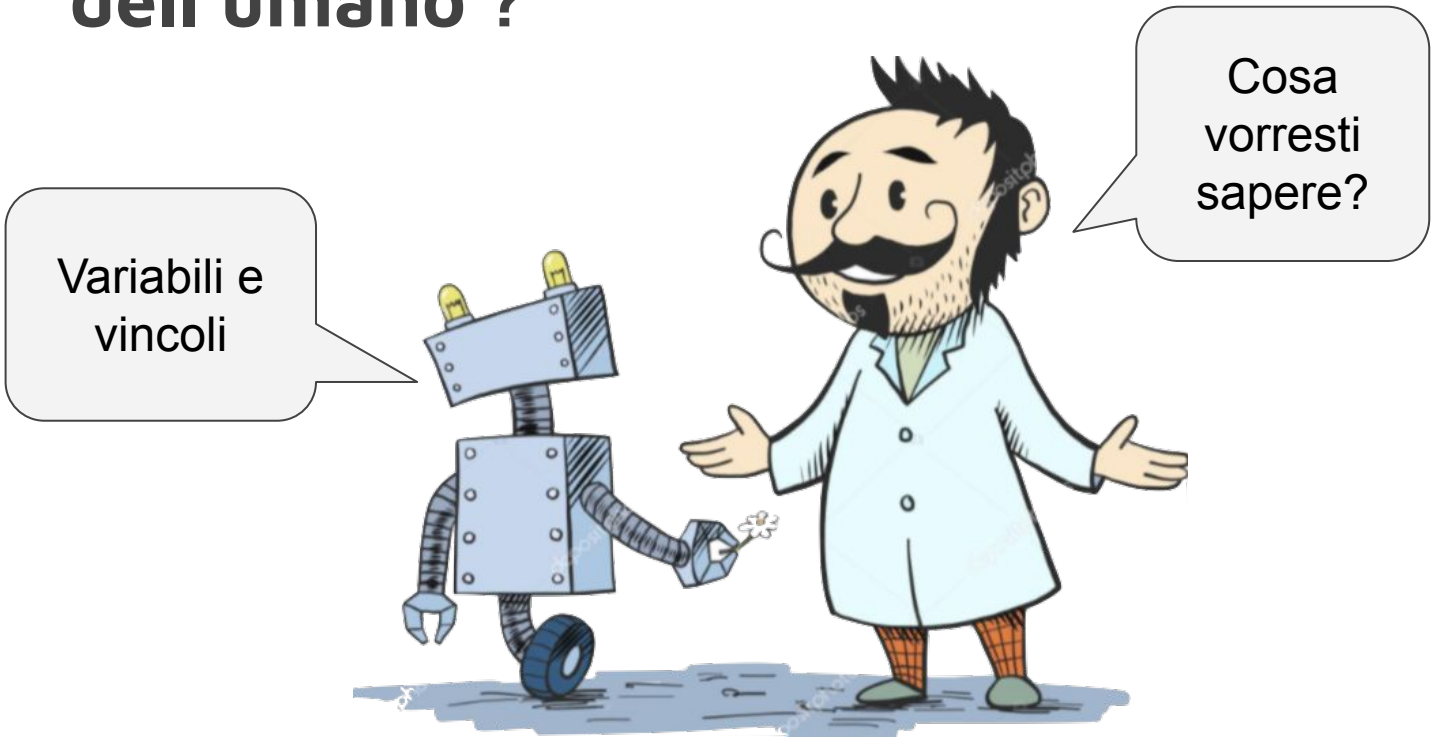
Descrivere il problema al computer

E.g. html, css. MiniZinc

Linguaggio dichiarativo

- Possiamo:
  - Non sapere l’esistenza della soluzione
  - Non sapere come ottenere la soluzione
- Ma dobbiamo:
  - Descrivere il problema

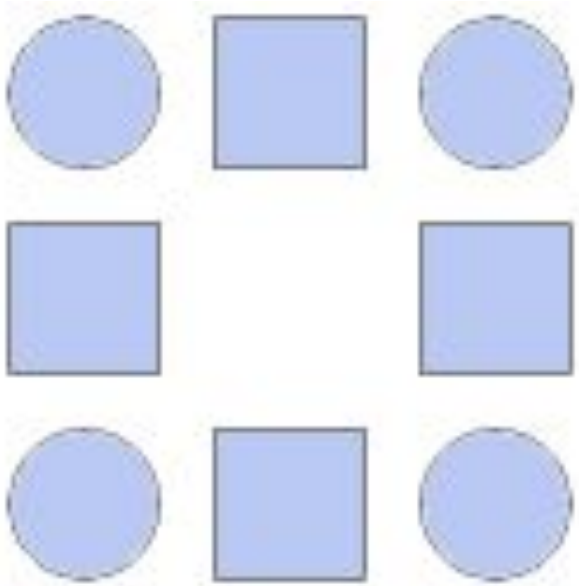
Come il computer comprende i desideri dell'umano ?



Cosa vogliono sapere i computer ?

- Variabili (entità, oggetti) del problema, quali sono
  - Tipi: numeri, insieme di numeri
  - Dominio
  - E.g. x in [1,10], y in [1,10]
- Vincoli (Relazioni) fra le entità
  - E.g. x != y

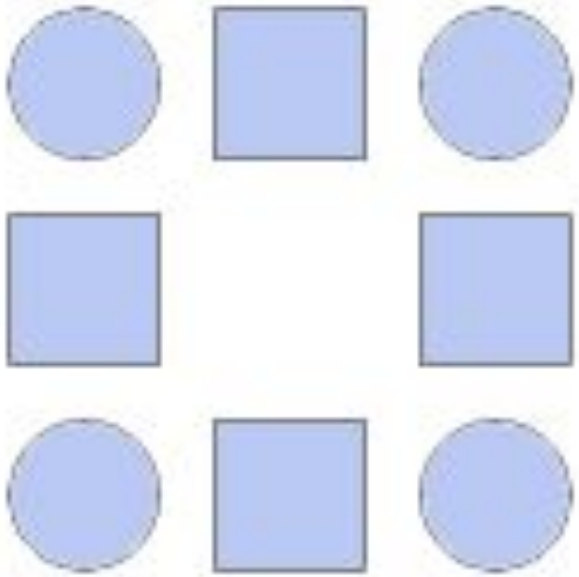
Added Corner Problem



Riempire le caselle con numeri da 1 a 8 (non ripetuti)
assicurando che il numero in quadro sia uguale alla somma dei due numeri adiacenti in cerchio.

.

Added Corner Problem

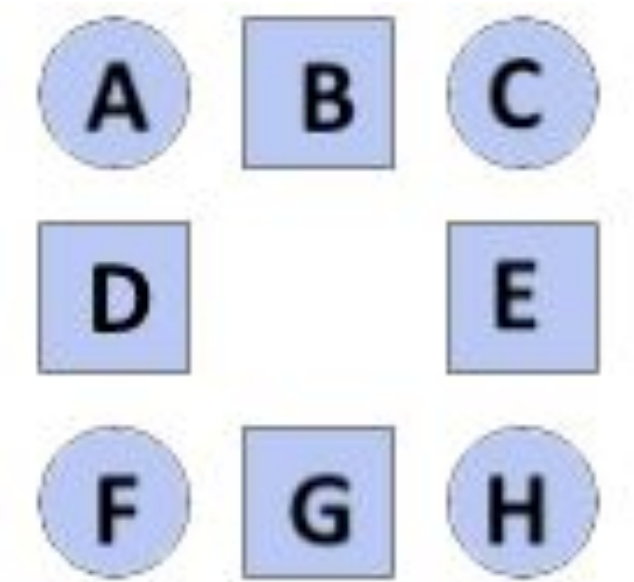


Quante variabili ?

Qual’e’ il dominio ?

Che relazioni ci sono fra loro ?

Added Corner Problem



Quante variabili ?

- 8

Qual’e’ il dominio ?

- 1..8

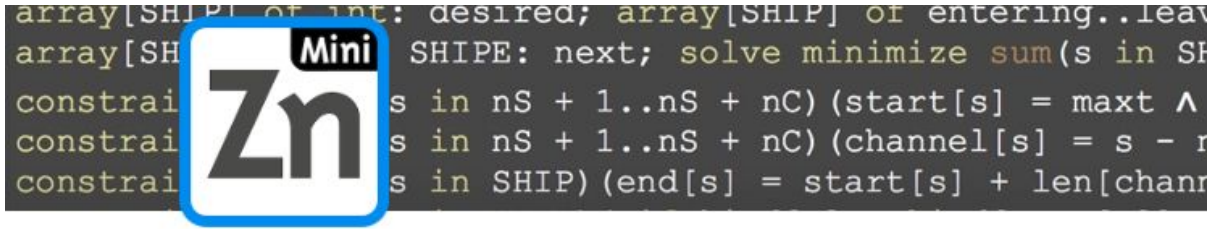
Che relazioni ci sono fra loro ?

- Regola del gioco

Codificare con Minizinc

Minizinc

- 2007 - oggi, University of Melbourne & Monash University, Australia
- Linguaggio di modellazione
- modello + istanza + Solver = Soluzione



Sintassi Minizinc

- ***Parametro*** rappresenta un'entita' che ha un valore fisso
  - [dominio] : [nome parametro]
- ***Variabile*** rappresenta un'entita' il cui valore puo' variare
  - var [dominio] : [nome parametro]

- Parametri, 8 caselle
  - int : num_caselle = 8;
  - set of int: valori = 1..8;
- Variabili , A, B, ..., H e Dominio 1..8
  - var 1..8 : A ;
  - var valori : B ;

Sintassi Minizinc

- Relazioni (vincoli)
  - Regole annotate con parole chiave ‘constraint’
  - constraint A != B, B != C, A != C
    - all_different(A, B, C) % vincolo globale
  - constraint B = A + C …
  - /\ operatore ‘and’
- Vincoli globali servono per vincolare piu’ variabili
- Le keyword constraint possono essere sostituite da operatori ‘and’

Come e’ fatto un sistema basato sulle regole?

- Campi di parametri e variabili
- Le regole espresse in forma di ‘constraints’

+

- Librerie da chiamare
- Obiettivo del sistema
- Output (stampa)

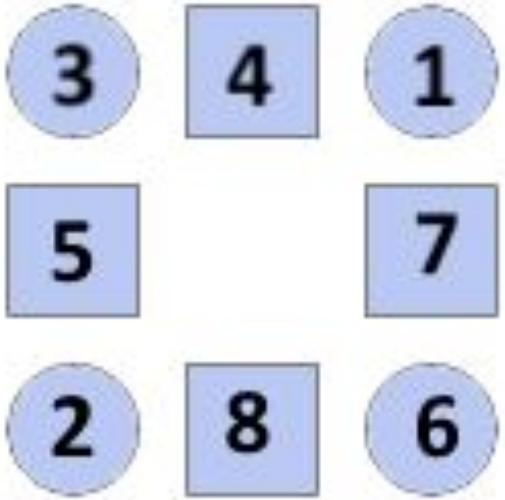
Modellazione in Minizinc

```
1  include "alldifferent.mzn";
2
3  var 1..8: A; var 1..8: B; var 1..8: C; var 1..8: D;
4  var 1..8: E; var 1..8: F; var 1..8: G; var 1..8: H;
5
6  constraint
7    all_different([A,B,C,D,E,F,G,H]);
8
9  constraint
10   B = A + C  /\ D = A + F  /\  E = C + H  /\ G = F + H;
11
12 solve satisfy;
13
14 output [
15   show(A), " ", show(B), " ", show(C), "\n",
16   show(D), "   ",  show(E), "\n",
17   show(F), " ", show(G), " ", show(H),"\n"
18 ];
19
```

Per potere usare all_different

Specificare l'obiettivo

Per stampare i risultati

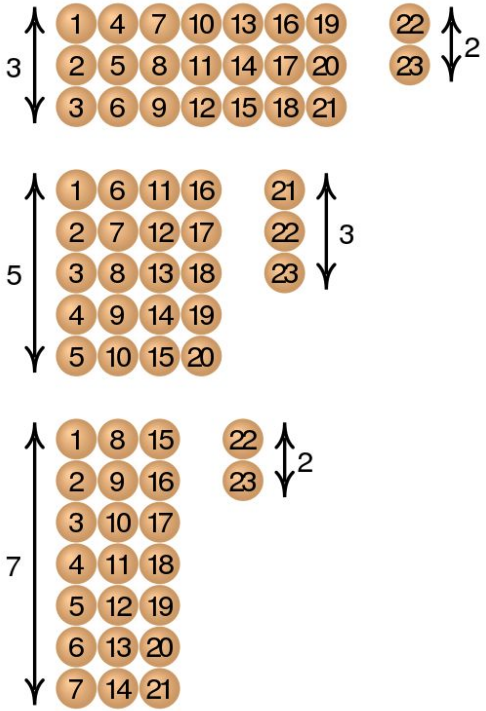


Domande

- Esiste una soluzione se il dominio e’ da 9 a 17 ?
- Esiste una soluzione se la somma di B D E G (numeri in quadro) fa 100 ? senno’ come posso fare ?

Giochi di Puzzle

Teorema cinese del resto (200 AC.)



Sappiamo che,
Per 23 elementi, facendo la fila in modi diversi,
nell'ultima fila restano diversi numeri di elementi.

Domanda

Ci sono 350 soldati all’inizio, alcuni sono scappati. Quando stanno in fila da 3 ne rimangono 2, facendo la fila da 5 persone, ne restano 4, se ciascuna fila contiene 7 persone ne restano 6.

Quanti soldati ci sono adesso?

Soluzione teorica

Si definisca il seguente sistema (con $MCD(n_1, n_2, n_3) = 1$):

$$\begin{cases} x \equiv a_1 \pmod{n_1} \\ x \equiv a_2 \pmod{n_2} \\ x \equiv a_3 \pmod{n_3} \end{cases}$$

Sia $N = n_1 \cdot n_2 \cdot n_3$

$$N_1 = n_2 n_3$$
$$N_2 = n_1 n_3$$
$$N_3 = n_1 n_2$$

Siano poi y_i le soluzioni alle congruenze $N_i y_i \equiv 1 \pmod{n_i}$; la soluzione sarà data da:

$$x \equiv a_1 N_1 y_1 + a_2 N_2 y_2 + a_3 N_3 y_3 \pmod{N}$$

Cenni

Soluzione teorica

Si definisca il seguente sistema (con $MCD(n_1, n_2, n_3) = 1$):

$$\begin{cases} x \equiv a_1 \pmod{n_1} \\ x \equiv a_2 \pmod{n_2} \\ x \equiv a_3 \pmod{n_3} \end{cases}$$

a1 = 2, n1 = 3
a2 = 4, n2 = 5
a3 = 6, n3 = 7

Sia $N = n_1 \cdot n_2 \cdot n_3$

N = 105

$$N_1 = n_2 n_3$$
$$N_2 = n_1 n_3$$
$$N_3 = n_1 n_2$$

N1 = 35
N2 = 21
N3 = 15

Siano poi y_i le soluzioni alle congruenze $N_i y_i \equiv 1 \pmod{n_i}$; la soluzione sarà data da:

$$x \equiv a_1 N_1 y_1 + a_2 N_2 y_2 + a_3 N_3 y_3$$

X = 2*35*2 + 4*21*1+ 6*15*1 = 314

Cenni

Modellazione con Minizinc

- Quante variabili ?
- Quali vincoli ci sono?

Soluzione con Minizinc

- Quante variabili ?
- Quali vincoli ci sono?

```
1 var 1..350: soldati;
2 constraint  soldati mod 3= 2;
3 constraint  soldati mod 5= 4;
4 constraint  soldati mod 7= 6;
5 solve satisfy ;
6 output [show(soldati)];
```

Fatto!

Cripto aritmetica

Che valori possono assumere le lettere?

- UNO + UNO = DUE
- Il dominio per U,N,O,E,D è fra 0 a 9

Suggerimenti:

Var 1..9 : U; var 0..9:N; var 0..9:O; etc

Fatto!

Più problemi

- ABCDE * F = GGGGGG
- GIOCO + RIDO = BELLO

Fatto!

Puzzle solving con Intelligenza Artificiale, parte 2

Dott. Tong Liu
Prof. Maurizio Gabbrielli
Alma Mater Studiorum - Università di Bologna

Ripasso

- Problema di soddisfacimento di vincoli
  - Constraint Satisfaction Problem
- Linguaggio Dichiarativo
  - descrivere/modellare il problema
- Esercizi in Minizinc
  - sintassi
  - componenti

Modellazione in Minizinc

```
1  include "alldifferent.mzn";
2
3  var 1..8: A; var 1..8: B; var 1..8: C; var 1..8: D;
4  var 1..8: E; var 1..8: F; var 1..8: G; var 1..8: H;
5
6  constraint
7    all_different([A,B,C,D,E,F,G,H]);
8
9  constraint
10   B = A + C  /\ D = A + F  /\  E = C + H  /\ G = F + H;
11
12 solve satisfy;
13
14 output [
15   show(A), " ", show(B), " ", show(C), "\n",
16   show(D), "   ",  show(E), "\n",
17   show(F), " ", show(G), " ", show(H),"\n"
18 ];
19
```

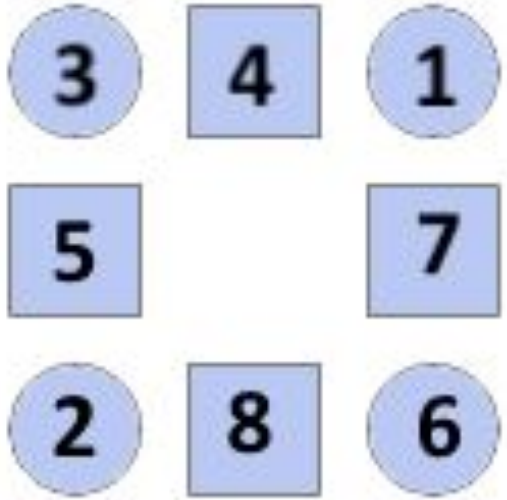
Includere la libreria

variabili

vincoli

L'obiettivo del problema

stampare i risultati

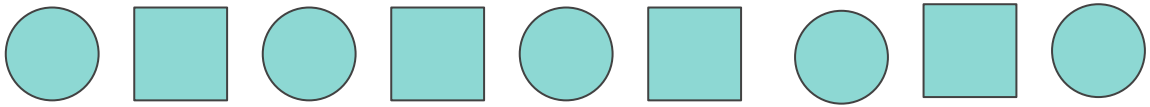


Scalare la potenza: `array` e Iterazione

Contenuti Parte 2

- Concetto di array e iterazione
- Array e iterazione in Minizinc
- Esercizi
  - Variante di Added Corner Problem
  - Sudoku
- Successi di CP in industria

Un altro problema

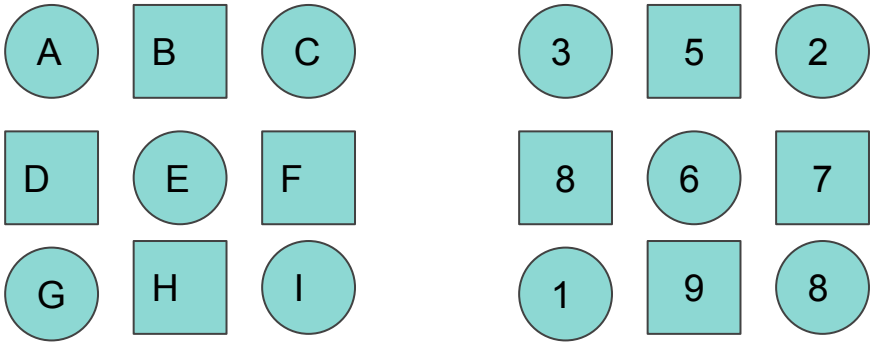


Vogliamo trovare 9 numeri distinti dove il numero in quadro e' la somma dei numeri adiacenti in cerchio.

Un altro problema

```
include "alldifferent.mzn";

var 1..9: A; var 1..9: B; var 1..9: C;
var 1..9: D; var 1..9: E; var 1..9: F;
var 1..9: G; var 1..9: H; var 1..9: I;

constraint
  all_different([A,B,C,D,E,F,G,H]);

constraint
  B = A + C  /\ D = C + E  /\
  F = E + G  /\ H = G + I;

solve satisfy;

output [
  show(A), " ", show(B), " ", show(C), "\n",
  show(D), " ",  show(E)," " ,  show(F),  "\n",
  show(G), " ", show(H), " ", show(I),"\n"
];
```

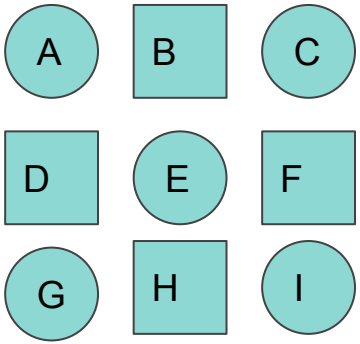
Una soluzione puo' essere il seguente

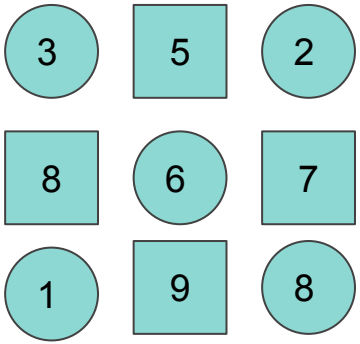


Un altro problema

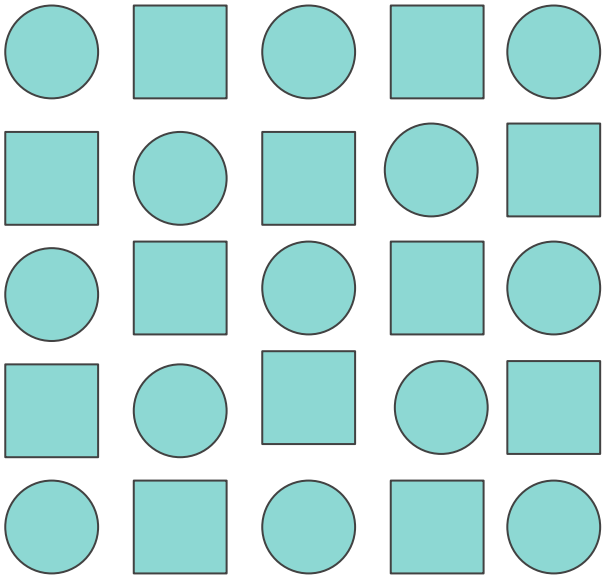
```
include "alldifferent.mzn";

var 1..9: A; var 1..9: B; var 1..9: C;
var 1..9: D; var 1..9: E; var 1..9: F;
var 1..9: G; var 1..9: H; var 1..9: I;

constraint
  all_different([A,B,C,D,E,F,G,H]);

constraint
  B = A + C  /\ D = C + E  /\
  F = E + G  /\ H = G + I;

solve satisfy;

output [
  show(A), " ", show(B), " ", show(C), "\n",
  show(D), " ",  show(E)," " ,  show(F),  "\n",
  show(G), " ", show(H), " ", show(I),"\n"
];

```

Pero' le nostre espressioni sono molto limitate!





Quando la dimensione aumenta ?

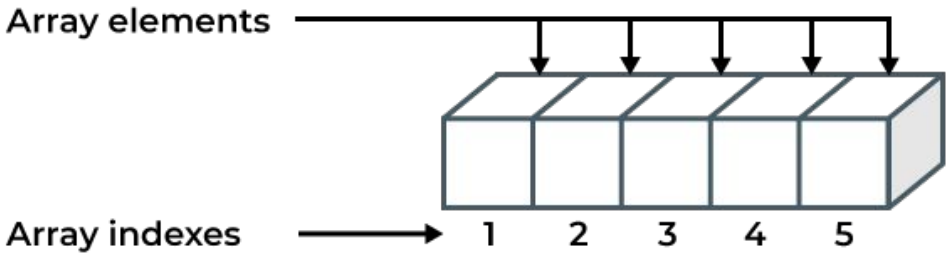


Cosa facciamo se la dimensione del problema e’ grande ?

Dobbiamo usare la nozione di ‘insieme’ - array

Array

Una collezione organizzata di oggetti

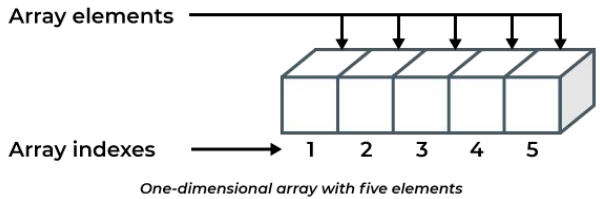


One-dimensional array with five elements

Sintassi Minizinc: array

- ***Array*** di parametri
  - array [1..indice] of int: [nome array]
- ***Array*** di variabili
  - array [1..indice] of var int: [nome array]

- E.g.
  - array [1..9] of var 1..9: caselle;

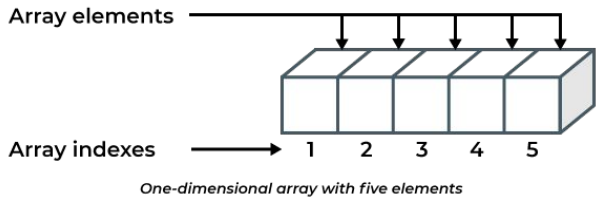


One-dimensional array with five elements

Sintassi Minizinc: iterazione

- Iteratore, a.k.a. cursore, contatore: un riferimento logico serve per visitare l'elemento di un insieme
- Contenuti di iterazione: cosa che facciamo con l'iteratore

```
constraint forall ( i in 1..5 )

                ( vincoli espressi con i);

constraint all_different(i in 1..5)(anArray[i]);
```



One-dimensional array with five elements

Sintassi Minizinc: esempio iterazione

Iteratore ‘i’

Condizione di ‘pari’

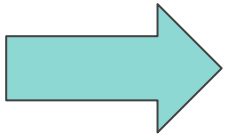
```
constraint forall(i in 1..9 where i mod 2 = 0)
    (caselle[i]=caselle[i-1]+caselle[i+1]);
```

Contenuto di iterazione

L’iterazione forall con l’obiettivo: elementi pari(in quadro) deve essere la somma dei due elementi adiacenti.

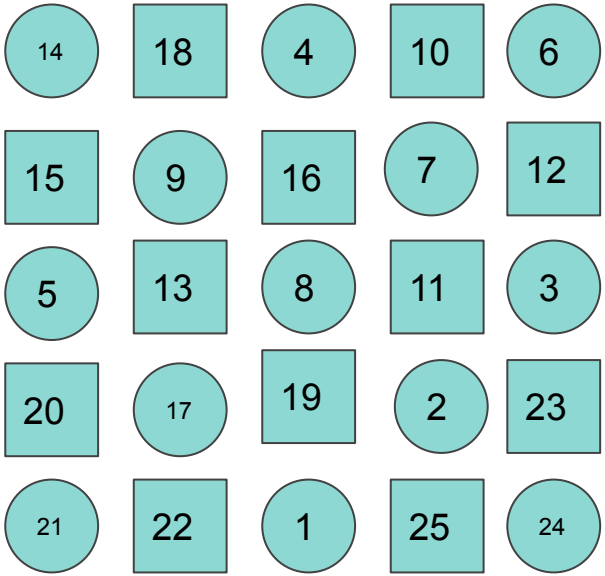
Modello nuovo

```
include "alldifferent.mzn";

var 1..9: A; var 1..9: B; var 1..9: C;
var 1..9: D; var 1..9: E; var 1..9: F;
var 1..9: G; var 1..9: H; var 1..9: I;

constraint
  all_different([A,B,C,D,E,F,G,H]);

constraint
  B = A + C  /\ D = C + E  /\
  F = E + G  /\ H = G + I;

solve satisfy;
```



```
include "alldifferent.mzn";

array [1..9] of var 1..9: caselle;
constraint all_different(caselle) ;

constraint forall(i in 1..9 where i mod 2 = 0)
    (caselle[i]=caselle[i-1]+caselle[i+1]);

solve satisfy;
```

Una possibile soluzione




```
1  include "alldifferent.mzn";
2
3  int : totale = 25;
4  array [1..totale] of var 1..totale: caselle;
5  constraint all_different(caselle) ;
6
7  constraint forall(i in 1..totale where i mod 2 = 0)
8     (caselle[i]=caselle[i-1]+caselle[i+1]);
9
10 solve satisfy;
```

Esempio Sudoku

Sudoku

7			3	4			1	
6	1	2		9		8		4
		3		6		2		
8	7	4	6	2	9	3	5	1
1	9	5	4	8	3	7	6	2
2	3	6	7	5	1	4	9	8
		9		1		5		
	6	1		7		9	2	3
4				3	5			6

Trovare numeri opportuni in modo tale che in ogni riga, ogni colonna e ogni regione quadrata con bordi in neretto, siano presenti tutti i numeri da 1 a 9 senza ripetizioni.

Mini Sudoku 2*2

	4	3	
	2	4	
	1		
	3		

Sudoku classico

7			3	4			1	
6	1	2		9		8		4
		3		6		2		
8	7	4	6	2	9	3	5	1
1	9	5	4	8	3	7	6	2
2	3	6	7	5	1	4	9	8
		9		1		5		
	6	1		7		9	2	3
4				3	5			6

Sfida impossibile 16*16

1						11	13			8	15			9	
		12	15	6				11			16	14	1	2	
	7			14				12	1			5		6	8
	8					2	4		3		14		11		15
9				1	3	12		13	14	6	4				
12		5	14		10	13				16		11	9		
10		8			16	14	2	3		5		6			
2		13				15					8	12		14	4
7	9		8	2					6				10		14
			6		12		5	14	16	1			2		3
		1	3		14				8	13		16	4		9
				16	8	10	1		4	3	11				5
8		11		7		3		4	12					5	
13	2		10			1	12				3			7	
	4	3	12	10			14				1	2	8		
	1			11	13			8	2						16

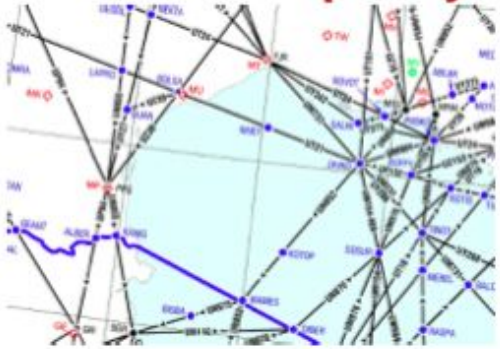
Sudoku soluzione

```
1  include "alldifferent.mzn";
2
3  int: S;
4  int: N = S * S;
5  array[1..N,1..N] of var 1..N: puzzle;
6
7
8  constraint
9     forall(i in 1..N)( alldifferent(j in 1..N)( puzzle[i,j] ))
10    /\
11    forall(j in 1..N)( alldifferent(i in 1..N)( puzzle[i,j] ))
12
13 constraint
14    forall(i,j in 1..S)
15       ( alldifferent(p,q in 1..S)( puzzle[S*(i-1)+p, S*(j-1)+q] ));
16
17 solve satisfy;
```

CP in industria

CP in industria

Demand vs capacity



Doctor rostering

Marketing campaign planning

cadence, AMPL, Boutiques, Google, Cisco, IBM, FICO, Jeppesen, NASA, Intel, Oracle, Quintiq, RedPrairie, Renault, SAP, Siemens, Tacton, Thales, Xerox, ...

Table plans for group conference

Robotic task sequencing

Plans for constructio n sites

Sports tournament design

Con Constraint Programming,
voi descrivete il problema,
l'AI pensa alla soluzione!

Crediti e riferimenti

- Pierre Flener,

  http://user.it.uu.se/~gusbj192/NordConsNet/nutCPshell-NordConsNet17.pdf
- Tong Liu, http://cs.unibo.it/~tong.liu3/mzn/slides_mzn.pdf
- Hakank, http://www.hakank.org/minizinc/#puzzles