

# N-Regine POSSIBILE SOLUZIONE IN PROLOG: GENERATE AND TEST

---

- La soluzione è rappresentata da una permutazione della lista: [1,2,3,4,5,6,7,8]
- La permutazione è una soluzione se le posizioni di ogni regina sono sicure.

```
solution(S) :- permutation([1, 2, 3, 4, 5, 6, 7, 8], S),  
            safe(S).  
permutation([], []).  
permutation([Head|Tail], PermList) :-  
    permutation(Tail, PermTail),  
    delete1(Head, PermList, PermTail).
```

(chiamando `delete1` di fatto si inserisce un elemento nella lista).

# N- regine con SOLUZIONE IN PROLOG: GENERATE AND TEST

---

- Definizione di safe:

- Se `S` è la lista vuota è sicuramente "safe".
  - Se `S` è una lista della forma `[Queen | Others]`, è safe se `Others` è safe e `Queen` non attacca nessuna regina in `Others`.

`safe([]).`

`safe([Queen | Others]) :-`

`safe(Others), noattack(Queen, Others, 1).`

`noattack(_, [], _).`

`noattack(Y, [Y1 | Ylist], Xdist) :-`

`Y - Y1 =\= Xdist,`

`Y1 - Y =\= Xdist,`

`Dist1 is Xdist + 1`

`noattack(Y, Ylist, Dist1).`

- Nota: questa impostazione non è particolarmente efficiente: genera una soluzione completa e poi la controlla (generate and test).

# PROBLEMA DELLE 8 REGINE

---

- Standard Backtracking in Prolog

```
solution(X) :-  
    queens(X, [], [1,2,3,4,5,6,7,8]).  
  
queens([], Placed, []).  
queens([X|Xs], Placed, Values) :-  
    delete1(X, Values, NewValues),  
    noattack(X, Placed),  
    queens(Xs, [X|Placed], NewValues).
```

- La **delete1**/3 istanzia la variabile **X** a un valore contenuto nel suo dominio.
- La **noattack**/2 controlla (a posteriori) che il valore scelto per **X** sia compatibile con le variabili già istanziate.

# FORWARD CHECKING IN PROLOG

---

```
queens([X1, X2, X3, X4, X5, X6, X7, X8], [1, 2, 3, 4, 5, 6, 7, 8]) :-  
    L = [1, 2, 3, 4, 5, 6, 7, 8],  
    queens_aux([[X1, L], [X2, L], [X3, L], [X4, L],  
               [X5, L], [X6, L], [X7, L], [X8, L]]).
```

```
queens_aux([]).
```

```
queens_aux([[X1, D] | Rest]) :-  
    member(X1, D), % istanzia la variabile X1  
    forward(X1, Rest, Newrest), % propagazione  
    queens_aux(Newrest).
```

```
forward(X, Rest, Newrest) :-  
    forward(X, Rest, 1, Newrest).
```

```
forward(X, [], Nb, []).  
// segue prossima
```

# FORWARD CHECKING IN PROLOG

---

```
forward(X, [[Var, Dom] | Rest], Nb, [[Var, [F | T]] | Newrest]) :-  
    remove_value(X, Dom, Nb, [F | T]),  
    Nb1 is Nb + 1,  
    forward(X, Rest, Nb1, Newrest).  
  
remove_value(X, [], Nb, []).  
remove_value(X, [Val | Rest], Nb, [Val | Newrest]) :-  
    compatible(X, Val, Nb), !,  
    remove_value(X, Rest, Nb, Newrest).  
remove_value(X, [Val | Rest], Nb, Newrest) :-  
    remove_value(X, Rest, Nb, Newrest).  
  
compatible(Value1, Value2, Nb) :-  
    Value1 =\= Value2 + Nb,  
    Value1 =\= Value2 - Nb,  
    Value1 =\= Value2.
```

# I VINCOLI NEI LINGUAGGI

---

- Linguaggi di programmazione che combinano la dichiaratività della Programmazione Logica e l'efficienza della Risoluzione di Vincoli.
- Limitazioni della programmazione logica:
  - gli oggetti manipolati dai programmi logici sono strutture non interpretate per cui l'unificazione ha successo solo tra oggetti sintatticamente identici.
  - strategia di ricerca del tipo depth-first con backtracking cronologico nello spazio delle soluzioni e conducono a strategie del tipo Generate and Test.

# PROGRAMMAZIONE LOGICA A VINCOLI

---

- Programmazione logica a vincoli: Constraint Logic Programming CLP
- 1989 Jaffar Lassez
- CLP permette di:
  - associare a ciascun oggetto la sua semantica e le operazioni primitive che agiscono su di esso (domini di computazione quali reali, interi, razionali, booleani e domini finiti di ogni genere).
  - sfruttare procedure di ricerca nello spazio delle soluzioni più intelligenti che conducono ad una computazione guidata dai dati e ad uno sfruttamento attivo dei vincoli.

# LO SCHEMA CLP

---

- Sviluppato nel 1989 da Jaffar e Lassez.
- Aspetto chiave: l'aumento di flessibilità derivante dall'introduzione di oggetti semantici primitivi su cui il linguaggio può inferire.
- L'unificazione è solo un caso particolare di risoluzione di vincoli.
- Superamento di una delle lacune presenti nello schema tradizionale della PL che ne inficia il meccanismo fondamentale dell'unificazione.
- Due termini, intesi come strutture non interpretate, sono unificabili solo se sintatticamente identici.
- In tal modo la struttura 5 e la struttura 2+3 non sono considerate lo stesso oggetto.
- Predicato `is/2` che consente di rispondere affermativamente alla query `5 is 2+3` e alla query `A is 2+3` con `A=5, yes` ma non si comporta correttamente nel caso in si ponga la query `5 is A+3` rispondendo `A=2`.

## LO SCHEMA CLP (continua)

---

- Lo schema CLP(X) (dove X è il generico dominio di computazione) permette la definizione di oggetti semantici appartenenti ad X, di operazioni primitive e di relazioni (vincoli) su questi.
  - Estrarre dalle relazioni  $Y=2+3$  ,  $5=A+3$  le informazioni corrette sull'istanziazione della variabile libera.
  - Possibilità di trattare relazioni del tipo  $X+3=Y-2$ .
  - Questo vincolo viene mantenuto in forma implicita fino a quando una istanziazione di una delle due variabili libere fornisce informazioni sull'altra.
- Tra i principali domini di computazione per i quali è stato costruito un constraint solver troviamo i reali e lo schema CLP(R), i razionali con CLP(Q), gli interi CLP(Z), i booleani e i domini finiti.
- Domini come strumento realizzativo delle Tecniche di Consistenza.

# LO SCHEMA CLP

---

- Le Tecniche di Consistenza possono essere realizzate con la PL costruendo programmi strutturati in modo tale da garantire il *pruning* a priori dell'albero decisionale.
- Gli insiemi di valori ammissibili per ciascuna variabile possono essere trattati utilizzando le liste e alcune primitive per agire su di esse.
- Tuttavia le operazioni sulle liste sono molto pesanti computazionalmente.

# LO SCHEMA CLP

---

- Un algoritmo che realizza il Forward Checking con gli strumenti offerti dalla PL risulta, in media, meno efficiente di uno che utilizza lo Standard Backtracking per problemi di dimensione ridotta ( $n < 12$ ).
- Estendere la PL con meccanismi specifici per il trattamento delle Tecniche di Consistenza.
- Introdurre il concetto di *dominio* associato ad una variabile che ne definisca un *range* cioè un insieme di valori ammissibili.
- La realizzazione dei domini deve essere tale da permettere un loro utilizzo efficiente quindi deve essere svincolata dalle liste e dal loro trattamento.

# DOMINIO NELLA CLP

---

- Supponiamo di avere a disposizione una primitiva

`domain(X, D)` che associa alla variabile  $X$  un insieme di valori possibili contenuto in  $D$ , e una primitiva

`indomain(X)` che istanzia  $X$ , in modo backtrackabile, con un elemento del suo dominio.

- Un vincolo su una variabile con dominio agisce sul dominio stesso riducendolo.

# DOMINIO NELLA CLP

---

- Esempio  
  :- **domain(X, [1, 2, 3, 4, 5]), X>3.**
- ha come effetto la riduzione del dominio di **X** a **[4, 5]** mentre il vincolo **X>7** conduce al fallimento non essendoci, nel dominio **X**, alcun valore che soddisfa il vincolo imposto.
- Arricchendo la PL in questo modo è possibile esprimere relazioni tra le variabili con dominio:  
  :- **domain(X, [1, 2, 3, 4, 5]), domain(Y, [3, 4, 5, 6]), X=Y.**
- L'unificazione delle variabili con dominio deve essere opportunamente gestita e la query precedente effettua la riduzione dei domini di **X** e di **Y** ai soli valori **[3, 4, 5]**, intersezione dei due domini.

# N-Regine FORWARD CHECKING CON DOMINI

---

- Molto più leggibile ed efficiente

```
queens([]).
```

```
queens([X|Y]) :-
```

```
    indomain(X),
```

```
    noattack(X, Y),
```

```
    queens(Y).
```

```
noattack(X, Y) :-
```

```
    noattack(X, Y, 1).
```

```
noattack(X, [], Nb).
```

```
noattack(X, [Y|Ys], Nb) :-
```

```
    X ≠ Y,
```

```
    X ≠ Y-Nb,
```

```
    X ≠ Y+Nb,
```

```
    Nb1 = Nb+1,
```

```
    noattack(X, Ys, Nb1).
```

- I vincoli contenuti in `noattack/3` agiscono eliminando i valori dai domini di `Y` non consistenti.
- Chiaramente tutte le variabili devono essere dotate di dominio tramite la primitiva