Nonograma bidireccional
Pràctica de Programació Lògica

Paradigmes i Llenguatges de Programació

Curs 23/24

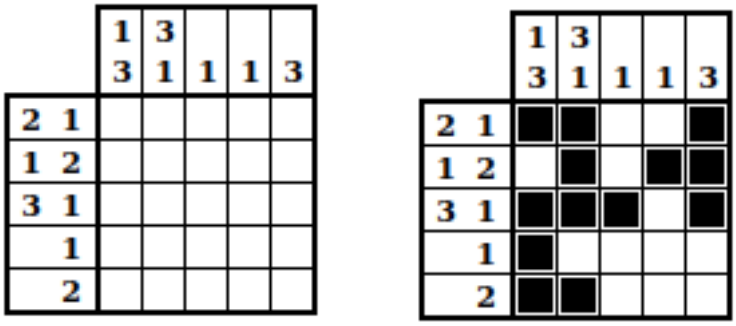

Figura 1: Exemple de nonograma buit a l'esquerra, i la seva solució a la dreta.

Introducció

Els nonogrames[1][2] són uns jocs de lògica en què hem de pintar un subconjunt de caselles d'una graella per tal de revelar un dibuix. Quines caselles s'han de pintar ve determinat per un conjunt d'indicacions per cada fila i per cada columna en forma de llista d'enters. Per exemple, la llista d'enters $[2, 4, 3]$ per una fila ens indica que, mirant les seves caselles d'esquerra a dreta, en algun lloc de la fila hi ha 2 caselles seguides pintades, seguides d'una o més caselles buides, seguides de 4 caselles seguides pintades, seguides d'una o més buides, i seguides de tres caselles pintades. Als extrems de la fila hi pot haver 0 o més caselles buides. Vist com a expressió regular on p significa ple i b significa buit, el patró $[2, 4, 3]$ correspon a $b^*p^2b^+p^4b^+p^3b^*$. Les indicacions per les columnes segueixen el mateix format, i s'han d'aplicar llegint les caselles de dalt a baix. Una llista buida [ ] significa que no hi ha cap casella plena en aquella fila/columna. La Figura 1 ens mostra un exemple de nonograma solucionat.

Un diari vol incloure els nonogrames a la seva secció de passatemps, i ens ha demanat un software per facilitar el procés de generació de problemes. Les funcionalitats del software han de ser molt més que un simple solucionador. Ens comenta el client que els casos d'ús habitual seran:

- Comprovar si els problemes tenen solució, i mostrar-la si és el cas.

[1] https://en.wikipedia.org/wiki/Nonogram
[2] https://es.puzzle-nonograms.com/

- Comprovar si el nombre de solucions és únic.
- Generar les indicacions de files i columnes a partir d'un dibuix.
- Fer una versió més difícil del joc, on es dona el dibuix parcialment pintat i les indicacions parcialment plenes, i el jugador ha d'esbrinar les indicacions que falten i acabar de pintar el dibuix. Això també serveix com a generador a partir de solucions parcials.

La programació lògica resulta un paradigma ideal per aquest cas. Per una banda, ens proporciona un motor de cerca, la SLD resolució, que ens estalvia haver d'implementar explícitament algorismes de backtracking. Per altra banda, la possibilitat d'instanciar un subconjunt de les dades i fer consultes sobre les variables no instanciades ens permetrà fer un sol programa que supleixi totes les funcionalitats del client mitjançant diferents consultes.

Què hem de fer

L'objectiu final és implementar un predicat que ens permeti completar un nonograma del qual sabrem, obligatòriament, les mides del tauler, i opcionalment, un subconjunt de les caselles pintades i un subconjunt de els indicacions. Per arribar a aquest objectiu final, es demanen les següents funcionalitats, cadascuna amb una contribució independent a la nota final.

Solucionador de nonograma (2.5 punts)

Fes un predicat `solucionarNono(+NF,+NC,+IF,+IC,G)` on:

NF i NC: són el nombre de files i columnes respectivament, es passen instanciades.

IF i IC: són les indicacions de les files i les columnes respectivament. IF és una llista que conté un nombre NF de subllistes, cada subllista **està instanciada**, i la i-èssima fila és la llista d'enters corresponent a les indicacions de la fila i. El contingut de IC és anàleg per les indicacions de les columnes.

G: és una llista de llistes, **no necessàriament instanciada**, que conté la solució del nonograma fila per fila. Una casella buida s'indica com un caràcter espai en blanc `' '`, i una casella pintada com un caràcter `'x'`. La primera fila de la solució de la Figura 1 seria `['x','x',' ',' ','x']`.

Exemples d'execució:

```
| ?- solucionarNono(2,2,[[2],[1]],[[2],[1]],G).
G = [[x,x],[x,' ']] ? ;
no
```

```
| ?- solucionarNono(3,3,[[1,1],[1],[1,1]],[[1,1],[1],[1,1]],G).
G = [[x,' ',x],[' ',x,' '],[x,' ',x]] ;
no

| ?- solucionarNono(5,5,[[1,2],[2,2],[1],[3,1],[1]],
    [[4],[1,1],[2],[2],[2,1]],G).
G = [[x,' ',' ',x,x],[x,x,' ',x,x],[x,' ',' ',' ',' '],
    [x,x,x,' ',x],[' ',' ',x,' ',' ']] ? ;
no
```

Generador de nonograma (2.5 punts)

Fes un predicat `generarNono(+NF,+NC,IF,IC,+G)` on:

NF i NC: són el nombre de files i columnes respectivament, es passen instanciades.

IF i IC: són les indicacions de les files i les columnes respectivament. IF és una llista **no necessàriament instanciada**. Per ser demostrat cert, IF ha de contenir un nombre NF de subllistes, on l'i-èssima és una llista d'enters corresponent a les indicacions de la fila i. El contingut de IC és anàleg per les indicacions de les columnes.

G: és una llista de llistes, **ja instanciada**, que conté la solució del nonograma fila per fila. Una casella buida s'indica com un caràcter espai en blanc `' '`, i una casella pintada com un caràcter `'x'`. La primera fila de la solució de la Figura 1 seria `['x','x',' ',' ','x']`.

Exemples d'execució:

```
| ?- generarNono(2,2,IF,IC,[['x','x'],['x',' ']]).
IC = [[2],[1]]
IF = [[2],[1]] ? ;
no

| ?- generarNono(3,3,IF,IC,[['x',' ','x'],[' ','x',' '],['x',' ','x']]).
IC = [[1,1],[1],[1,1]]
IF = [[1,1],[1],[1,1]] ? ;
no

| ?- generarNono(5,5,IF,IC,[['x',' ',' ','x','x'],['x','x',' ','x','x'],
    ['x',' ',' ',' ',' '],['x','x','x',' ','x'],[' ',' ','x',' ',' ']]).
IC = [[4],[1,1],[2],[2],[2,1]]
IF = [[1,2],[2,2],[1],[3,1],[1]] ? ;
no
```

Nonograma bidireccional (2.5 punts)

Fes un predicat `nonoBidirec(+NF,+NC,IF,IC,G)` on:

NF i NC: són el nombre de files i columnes respectivament, es passen instanciades.

IF i IC: són les indicacions de les files i les columnes respectivament. IF és una llista **no necessàriament instanciada**. Per ser demostrat cert, IF ha de contenir un nombre NF de subllistes, on l'i-èssima és una llista d'enters corresponent a les indicacions de la fila i. El contingut de IC és anàleg per les indicacions de les columnes.

G: és una llista de llistes, **no necessàriament instanciada**, que conté la solució del nonograma fila per fila. Una casella buida s'indica com un caràcter espai en blanc `' '`, i una casella pintada com un caràcter `'x'`. La primera fila de la solució de la Figura 1 seria `['x','x',' ',' ','x']`.

Si fas aquest predicat **no cal que implementis individualment els predicats `solucionarNono` i `generarNono`**. Encara que solucions especifiques pels dos predicats anteriors siguin més eficients, això no penalitzarà la nota i es replicarà la puntuació d'aquest aparat als dos apartats anteriors. Senzillament pots incloure:

```
solucionarNono(NF,NC,IF,IC,G):-nonoBidirec(NF,NC,IF,IC,G).
generarNono(NF,NC,IF,IC,G):-nonoBidirec(NF,NC,IF,IC,G).
```

Si no aconsegueixes completar correctament la implementació de `nonoBidirec`, es recomana implementar els altres dos separadament.

Exemple d'execució:

```
| ?- nonoBidirec(5,5,[IF1,[2,2],IF3,[3,IF4_2],[1]],[[4],[1,1],[2],[2],[2,1]],
    [['x',C22,' ','x','x'],['x','x',' ','x','x'],['x',' ',' ',' ',' '],
        ['x','x','x',' ','x'],F5]).
C22 = ' '
F5 = [' ',' ',x,' ',' ']
IF1 = [1,2]
IF3 = [1]
IF4_2 = 1 ? ;
no
```

Dibuixar nonograma(2.5 punts)

Fes un predicat que dibuixi un nonograma solucionat `nonoPrint(+NF,+NC,+IF,+IC,+G)` on totes les variables ja estan instanciades i corresponen a una solució correcta del nonograma. Integra'l amb cadascun dels tres predicats anteriors que hagis implementat, és a dir, fes també un `solucionarNonoPrint(+NF,+NC,+IF,+IC,G)`,

`generarNonoPrint(+NF,+NC,IF,IC,+G)` i `nonoBidirecPrint(+NF,+NC,IF,IC,G)`, que dibuixin el nonograma en acabar.

Es demana que l'aspecte visual de la graella sigui **exactament** el següent, fixeu-vos que entre columnes sempre hi ha un espai en blanc, llevat d'abans i després dels '|'. Si feu exemples amb pistes de més d'una xifra, es valorarà positivament que la graella quedi ben alineada.

```
| ?- solucionarNonoPrint(2,2,[[2],[1]],[[2],[1]],_).
 |2 1
-----
2|x x
1|x
true ? ;
no

| ?- generarNonoPrint(3,3,_,_,[['x',' ','x'],[' ','x',' '],['x',' ','x']]).
   |1   1
   |1 1 1
---------
1 1|x   x
  1|  x
1 1|x   x
true ? ;
no

| ?- nonoBidirecPrint(5,5,[_,[2,2],_,[3,_],[1]],[[4],[1,1],[2],[2],[2,1]],
        [['x',_,' ','x','x'],['x','x',' ','x','x'],['x',' ',' ',' ',' '],
        ['x','x','x',' ','x'],_]).
   |  1     2
   |4 1 2 2 1
-------------
1 2|x     x x
2 2|x x   x x
  1|x
3 1|x x x   x
  1|    x
true ? ;
no

?- nonoBidirecPrint(3,3,[[2],[2],[2]],_,_).
 |3 3
-------
2|x x
2|x x
```

```
2|x x
true ? a
 |2 3 1
-------
2|x x
2|x x
2|  x x
true
 |1
 |1 3 1
-------
2|x x
2|  x x
2|x x
true

 |1 3 2
-------
2|x x
2|  x x
2|  x x
true
 |2 3 1
-------
2|  x x
2|x x
2|x x
true
 |    1
 |1 3 1
-------
2|  x x
2|x x
2|  x x
true
 |1 3 2
-------
2|  x x
2|  x x
2|x x
true
 |  3 3
-------
2|  x x
```

```
2|  x x
2|  x x
true
no
```

Instruccions addicionals

El codi ha de complir les següents restriccions, i la violació de qualsevol d'elles comportarà **suspendre la pràctica amb un 0**.

- El nom i paràmetres dels predicats han de ser **exactament** els proposats als apartats anteriors.
- El codi ha de ser executable amb **GNU Prolog**, no s'acceptarà encara que sigui correcte amb altres implementacions com SWI-prolog o SICStus prolog.
- D'entre els predicats integrats, només es poden fer servir predicats i operadors aritmètics (*is*, operadors, *max*, *min*, ...), el *findall*, el *between*, els d'entrada/sortida, i tots els inclosos a l'apartat *List processing* de la documentació de GNU Prolog[3]. **Qualsevol altre predicat que volgueu utilitzar, l'haureu d'implementar.** En cas de dubte sobre si un predicat es pot fer servir, no dubteu a preguntar.
- El treball ha de ser 100% original. No es poden reutilitzar solucions totals ni parcials d'internet. En cas de plagi entre alumnes, també suspendran els autors originals.

Disposeu d'un fitxer predicatsRecomanats.pl on es suggereixen diferents predicats per implementar que us poden ser útils per fer la pràctica. Utilitzar-los és opcional.

Lliurament

- Les pràctiques **S'HAN** de fer en equips de dos.
- Cal que documenteu el codi, dient quins paràmetres té cada predicat, quin significat tenen, i si han d'estar instanciats, ja sigui total o parcialment (per exemple, podem demanar que una llista tingui la llargada definida però no el seu contingut).
- S'haurà lliurar al Moodle un fitxer `NomCognom1-NomCognom2.zip`, que contingui:
  - El fitxer del codi, anomenat `nonograma.pl`. Ha d'estar ben indentat i comentat.
  - Un fitxer `proves.txt` que contingui les consultes que heu fet per provar el programa.
  - Un PDF de màxim una pàgina que contingui:

[3] http://www.gprolog.org/manual/html_node/gprolog024.html

    1. Abast de la vostra solució: quins dels apartats heu implementat, i les possibles extensions o limitacions de la vostra solució.
    2. Referències de bibliografia i/o llibreries usades si s'escau.

- Molt probablement també hi haurà **entregues presencials** per poder avaluar la implicació en la pràctica per part dels membres de l'equip. És possible que per acabar d'avaluar se us faci fer alguna modificació de la pràctica *in situ*.

- Data de lliurament: **15 d'abril.**