1 Lexicographic order constraint

Question 1.1
Write a non-deterministic CLP program `lexleq(L1, L2)` enforcing the lexical order constraint between two tuples represented as lists with the same number of FD variables, using only constraints given above.

Question 1.2
Write the first two iterations of the $T_p$ operator on the above program.

Question 1.3
Write a deterministic CLP program enforcing the lexical order constraint between two lists of FD variables.

Question 1.4
Give an example where the second program (the deterministic one) propagates strictly more than the first one (assuming arc-consistency on the elementary constraints).
Suppose now again that only FD constraints are given, but this time in a CC(FD) concurrent constraint programming language, i.e., tells and asks are given for $X \neq Y + Z$, $X \neq Y + Z$, $X \leq Y + Z$, $X \geq Y + Z$ and their counterparts without $Z$.

When there is no ambiguity, tells can be omitted, but with careful parenthesizing.

For the reasons illustrated in the previous section, we want to add to our language reified constraints of the form:

$$B \# < \equiv \geq X \leq Y$$

where $B$ is a boolean variable with domain $\{0, 1\}$, and $c$ is one of the above FD constraints. The reified constraint enforces that the boolean $B$ equals 1 if and only if the constraint $c$ is satisfied.

Question 2.1
Write a deterministic (without the choice operator $+$) CC(FD) program $P$ defining the constraint $B \#<\equiv X \geq Y$ as a three-argument predicate, for the $O_{ts}$ observable.

Question 2.2
Give a denotational semantics $J P; B \#<\equiv X \geq Y$ of the constraint defined above, related to the $O_{ts}$ observable. Recall precisely how these two semantics are linked.

Note that it will not be necessary to decompose the sets $\uparrow c$ and $\downarrow c$ for the FD constraints considered, but that possible simplifications should be done.

Question 2.3
Write a non-deterministic version of program $P$, using $+$. What can you say about its observables “of interest”??
3  LCC definition and semantics

d\langle \text{c} \rangle \leftarrow \text{j} \text{r}_p \text{Hij} \_ \text{Lij} \_ \text{Oij} \_ \text{n} \_ \text{Sn} \_ \text{hq} \_ \text{q} \_ \text{O} \_ \text{Sij} \_ \text{Uij} \_ \text{Oij} \_ \text{Qij} \_ \text{B} \_ \#<\#> \_ \text{Y} \\
\text{domain}(X; \text{Min}; \text{Max}) = \text{tell}(\text{lb}(X; \text{Min})) \| \text{tell}(\text{ub}(X; \text{Max}))

Question 3.1

Write the predicate \text{geq}_x(X, Y) corresponding to the action on X of the \text{Soh} for X \#>=Y.

Question 3.2

Supposing that the above \text{Soh} is given, write the \text{geq}(B, X, Y) program corresponding to B \#<\#>\#>Y in LCC(H).

Question 3.3

Give a sound logical semantics in LJ of LCC agents, and prove its soundness.

Give an example of non-completeness of that semantics with respect to an observable of your choice.
4 Internalizing search as reified constraints

We have seen briefly in class that any search strategy can be internalized as constraint propagation and labelling over boolean variables and reified constraints. Here is an SWI-Prolog program that corresponds to the CLPZinc encoding of a dichotomic search of a variable between two bounds.

```prolog
:- use_module(library(clpfd)).

dichotomic_search(X, Min, Max) :-
    X in Min..Max,
    MaxIter is ceiling(log(Max - Min + 1)/log(2)),
    dichotomic_search_rec(X, L, MaxIter),
    label_and_indexicals(X, L).

dichotomic_search_rec(_, [], 0).

dichotomic_search_rec(X, [MinX, MaxX, B | L], N) :-
    N > 0,
    % in FD expressions / is the Euclidean division
    B #=<=> X #> (MinX + MaxX)/2,
    M is N - 1,
    dichotomic_search_rec(X, L, M).

label_and_indexicals(_, []).

label_and_indexicals(X, [Min, Max, B | L]) :-
    % fd_inf and fd_sup are indexicals on the current
    % min and max values in the domain of an FD variable
    fd_inf(X, Min),
    fd_sup(X, Max),
    ( var(B) % B not instantiated yet
    ->
    format("\~w\=\<\~w\=\<\~w\~n", [Min, Max]),
    label([B]),
    label_and_indexicals(X, L)
    ;
    true
    ).
```

**Question 4.1**

*Draw the search trees corresponding to the execution of the goals:*

- `dichotomic_search(X, 0, 4).`
- `X #\=2, dichotomic_search(X, 0, 4).`

**Question 4.2**

*Give an example where the use of equivalence constraints \( B \#<=> c \) allows for a better propagation than the usual strategy-directed search that corresponds to \( B \#=> c \).*