In Prolog, one first has to say which constraint theory CT should be used. CLP-libraries come in modules (some of them are included in Prolog-distributions).

`mse_module/1` is a predicate to import modules.

To import the library with the constraint theory CT, the Prolog program must contain the directive:

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

Problem: Constraint solver for CT is needed to check satisfiability of constraints in each computation step (and to simplify constraints).

=> Should be automatic + efficient.

But: most constraint theories are undecidable or only have very time-consuming decision procedures (CT+FD is undecidable).
Solution: Instead of checking 
\[ CT \cup \{ \forall X = X \}, \text{true} \] 
this question is only "approximated". This is efficient, but not always correct (i.e., there might be conjunctions of constraints \( \phi \) where Prolog falsely detects their satisfiability).

To approximate satisfiability in \( CT_{FD} \), one typically uses path-consistency.

**Def. 6.2.1 (Path Consistency)**

Let \( \phi_0 = \phi_1 \land \ldots \land \phi_m \) be a conjunction of constraints with \( \phi_i \in \text{At}(\Sigma_{FD}, \Delta_{FD}, \emptyset) \). Let \( X_1, \ldots, X_n \) be the variables in \( \phi_0 \) and let \( D_1, \ldots, D_n \) be subsets of \( \mathbb{Z} \). \( D_1, \ldots, D_n \) are admissible domains for \( X_1, \ldots, X_n \) w.r.t. \( \phi_0 \) iff for all \( \phi_i \) and all variables \( X_j \) the following holds:

for all \( a_j \in D_j \) there exist \( a_1 \in D_1, \ldots, a_{j-1} \in D_{j-1}, a_{j+1} \in D_{j+1}, \ldots, a_n \in D_n \) such that 
\[ CT_{FD} = \{ \phi_i \mid X_i / a_i, \ldots, X_n / a_n \} \]
\( \phi_0 \) is path-consistent iff there are admissible domains
D1, ..., Dn that are all not empty.

Problem: Satisfiability of the constraints separately:
if φ1 and φ2 are both satisfiable,
then φ1 ∨ φ2 still does not need to be satisfiable.

Automated checking of pair consistency:
1. Let D1 = Z, ..., Dn = Z.
2. Process the constraints φ after each other. For each φ:
3. Process the variables after each other. For xj:
   Reduce its domain to those values where φ can be made true if the other variables can only take values from their domains.
4. The whole process is repeated until the constraints do not change anymore.

**Example**

Let C0 be

\[ x_1 \#> 5 \land x_1 \#< x_2 \land x_2 \#< 9 \]

- **Beginning:** D1 = Z, D2 = Z
- **Consider:** x1 > 5 \[\Rightarrow D_1 = \{6,7,8,\ldots\}\], D2 = Z
- **Consider:** x1 \#< x2 \[\Rightarrow\]
\[ D_1 = \{6, 7, 8, \ldots\}, \quad D_2 = \mathbb{Z} \quad \text{For every } a_1 \in D_1 \]
\[ \text{there exists } a_2 \in D_2 \quad \text{such that } a_1 < a_2. \]

\[ D_1 = \{6, 7, \ldots\}, \quad D_2 = \{7, 8, \ldots\} \quad \text{For every } a_2 \in D_2 \]
\[ \text{there exists } a_1 \in D_1 \quad \text{such that } a_1 < a_2. \]

- Consider \( x_2 < 9 \) \( \Rightarrow \) \( D_1 = \{6, 7, \ldots\}, \quad D_2 = \{7, 8\} \)

- Consider \( x_1 > 5 \) \( \Rightarrow \) \( D_1 = \{6, 7, \ldots\}, \quad D_2 = \{7, 8\} \)

- Consider \( x_1 < x_2 \) \( \Rightarrow \) \( D_1 = \{6, 7\}, \quad D_2 = \{7, 8\} \)

Now nothing changes anymore \( \Rightarrow \)
\[ C_0 \text{ is part-consistent } (D_1 \neq \emptyset, D_2 \neq \emptyset). \]

Simplify \( C(0) = \) \( x_1 \) in \( 6..7 \), \( x_1 < x_2 \), \( x_2 \) in \( 7..8 \)

The constraint signature contains more symbols:

\[ \uparrow \quad \uparrow \quad \uparrow \]

\( \uparrow \infty \quad \uparrow \infty \)

CT is “should be used for finite domains, but this is not enforced”:
\[ ?= x_1 \text{ in } 6..\sup, \quad x_1 < x_2, \quad x_2 \text{ in } \inf..8 \]

\[ \uparrow \quad \uparrow \]

\[ \uparrow \infty \quad \uparrow \infty \]
CT has a predicate `label` which enforces that all solutions are enumerated:

![- X₁ ≠ 5, X₁ ≤ X₂, X₂ ≤ 9, label ([X₁, X₂])

```
X₁ = 6, X₂ = 7 ;
X₁ = 7, X₂ = 8 ;
false
```

`label` can only be used if all the variables have finite domains:

?- X₁ ≠ 5, X₁ ≤ X₂, label ([X₁, X₂]).

pred: error

**Example 6.2.3** Incorrectness of Path Consistency

?- X₁ ≠ X₂, X₁ ≤ X₂.

X₁ ≠ X₂ ∧ X₁ ≤ X₂ is path-consistent, but unsatisfiable.

D₁ = Z    D₂ = Z are admissible domains.
for every $a_n \in D_1$, there exists $a_2 \in D_2$ such that $a_n \neq a_2$

- for every $a_n \in D_1$, there exists $a_2 \in D_2$ such that $a_n \neq a_2$

- for every $a_2 \in D_2$, ...

**Ex 6.2.4  N-queens problem**

- chess board of size $n \times n$
- place $n$ queens on the board that cannot beat each other

```
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
1 & & & x \\
2 & x & & \\
3 & & x & \\
4 & & & x \\
\end{array}
```

- Represent the positions of queens by a list $[x_1, \ldots, x_n]$ where $x_i$ is the row for the queen of column $i$. (e.g. $[2, 4, 1, 3]$).
- $\mathcal{Q}(4, L)$ will compute solution $L$ for chess-board of size $4 \times 4$.
- "$L \text{ ins } 1..N$" means "$X \text{ in } 1..N$" for
every element $X$ of $L$

- all different is pre-defined
- first three literals: $\leq$ is a permutation of \{1, .., $N$\}