NOPT042 Constraint programming: Tutorial 5 – Advanced constraint modelling

What was in the lecture? Nothing, the lecture was canceled.

The element constraint

Picat doesn't support indexation in constraints, e.g. `X #= L[I]`. Instead, it implements the `element` constraint:

```
element(I, L, X)
```

("X is on the Ith position in the list L"). But this constraint can also be used for "reverse indexing": when we have X and want to find its position in L. The constraint doesn't care about the direction; this is called *bidirectionality*.

For matrices (2D arrays), `Matrix[I,J]#=X` is expressed using the following constraint:

```
matrix_element(Matrix,I,J,X)
```

In [1]:
```
%load_ext ipicat
```

```
Picat version 3.7
```

In [2]:
```
%%picat
import cp.

main =>
    L = new_array(10),
    L :: 0..9,
    all_different(L),
    % X #= L[I], % this does not work, use element instead
    element(I, L, X),
    X #= 5,
    solve($[max(I)],L),
    println(L).
```

```
{0,1,2,3,4,6,7,8,9,5}
```

Exercise: Langford's number problem

Consider the following problem (see the book):

> Consider two sets of the numbers from 1 to N. The problem is to arrange the $2N$ numbers in the two sets into a single sequence in which the two 1's appear one number apart, the two 2's appear two numbers apart, the two 3's appear three numbers apart, etc.

Try to formulate a model for this problem.

In [3]:
```
!ls exercises/langford/.solution/
```

```
langford-channeling.pi  langford-primal-search-strategy.pi
langford-dual.pi        langford-primal.pi
```

In [4]:
```
!picat exercises/langford/.solution/langford-primal 8
```

```
Default search strategy:{1,3,1,6,7,3,8,5,2,4,6,2,7,5,4,8}
```

In [5]:
```
!picat exercises/langford/.solution/langford-primal-search-strategy 8
```

```
Good search strategy:{2,4,7,2,8,6,4,1,5,1,7,3,6,8,5,3}
```

In [6]:
```
!picat exercises/langford/.solution/langford-dual 8
```

```
Positions: {1,4,8,11,9,6,2,5,3,7,12,16,15,13,10,14}
```

In [7]:
```
!picat exercises/langford/.solution/langford-channeling 8
```

```
CPU time 3.073 seconds. Backtracks: 250746

{1,3,1,6,7,3,8,5,2,4,6,2,7,5,4,8}
Positions: {1,9,2,10,8,4,5,7,3,12,6,15,14,11,13,16}
```

In [8]:
```
!picat exercises/langford/.solution/langford-dual 20
```

```
Positions: {1,2,4,6,7,9,19,20,18,21,24,25,26,22,23,14,17,15,10,12,3,5,8,11,1
3,16,27,29,28,32,36,38,40,37,39,31,35,34,30,33}
```

The assignment problem

From Wikipedia:

The assignment problem is a fundamental combinatorial optimization problem. In its most general form, the problem is as follows:

> The problem instance has a number of agents and a number of tasks. Any agent can be assigned to perform any task, incurring some cost that may vary depending on the agent-task assignment. It is required to perform as many tasks as possible by assigning at most one agent to each task and at most one task to each agent, in such a way that the total cost of the assignment is minimized.

Alternatively, describing the problem using graph theory:

> The assignment problem consists of finding, in a weighted bipartite graph, a matching of a given size, in which the sum of weights of the edges is minimum.

If the numbers of agents and tasks are equal, then the problem is called balanced assignment.

Exercise: Swimmers

(From: W. Winston, Operations Research: Applications & Algorithms.)

In medley swimming relay, a team of four swimmers must swim 4x100m, each swimmer using a different style: breaststroke, backstroke, butterfly, or freestyle. The table below gives their average times for 100m in each style. Which swimmer should swim which stroke to minimize total time?

Swimmer	Free	Breast	Fly	Back
A	54	54	51	53
B	51	57	52	52
C	50	53	54	56
D	56	54	55	53

Write a general model, generate larger instances, and try to make your model as efficient as possible.

In [9]:
```
!picat exercises/swimmers/instances
```

```
Sample instance:
{A,B,C,D}
{Free,Breast,Fly,Back}
{{54,54,51,53},{51,57,52,52},{50,53,54,56},{56,54,55,53}}

Random instance:
{Swimmer1,Swimmer2,Swimmer3,Swimmer4,Swimmer5,Swimmer6,Swimmer7,Swimmer8}
{Style1,Style2,Style3,Style4,Style5,Style6,Style7,Style8}
{{46,49,54,64,53,55,55,54},{60,55,47,64,65,49,65,52},{48,60,61,61,62,59,57,5
4},{47,50,50,58,46,64,50,45},{48,57,62,54,46,52,61,61},{60,63,57,59,65,55,6
5,47},{47,60,62,64,52,53,53,54},{56,56,46,55,54,51,56,57}}
```

In [10]:
```
!picat exercises/swimmers/.solution/swimmers primal
!picat exercises/swimmers/.solution/swimmers dual
!picat exercises/swimmers/.solution/swimmers channeling
```

```
# primal model:
Total time: 207
Swimmer A is swims Fly
Swimmer B is swims Back
Swimmer C is swims Free
Swimmer D is swims Breast
# dual model:
Total time: 207
Style Free is swum  by C
Style Breast is swum  by D
Style Fly is swum  by A
Style Back is swum  by B
# channeling model:
Total time: 207
Swimmer A is swims Fly
Swimmer B is swims Back
Swimmer C is swims Free
Swimmer D is swims Breast
or in the dual view
Style Free is swum  by C
Style Breast is swum  by D
Style Fly is swum  by A
Style Back is swum  by B
```

In [11]:
```
!time picat exercises/swimmers/.solution/swimmers primal random
!time picat exercises/swimmers/.solution/swimmers dual random
!time picat exercises/swimmers/.solution/swimmers channeling random
```

```
# primal model:
Total time: 848
Swimmer Swimmer1 is swims Style3
Swimmer Swimmer2 is swims Style6
Swimmer Swimmer3 is swims Style16
Swimmer Swimmer4 is swims Style1
Swimmer Swimmer5 is swims Style14
Swimmer Swimmer6 is swims Style7
Swimmer Swimmer7 is swims Style13
Swimmer Swimmer8 is swims Style19
Swimmer Swimmer9 is swims Style5
Swimmer Swimmer10 is swims Style10
Swimmer Swimmer11 is swims Style12
Swimmer Swimmer12 is swims Style9
Swimmer Swimmer13 is swims Style4
Swimmer Swimmer14 is swims Style11
Swimmer Swimmer15 is swims Style8
Swimmer Swimmer16 is swims Style17
Swimmer Swimmer17 is swims Style2
Swimmer Swimmer18 is swims Style18
Swimmer Swimmer19 is swims Style15
Swimmer Swimmer20 is swims Style20

real    0m0.032s
user    0m0.026s
sys     0m0.000s
```

```
# dual model:
Total time: 848
Style Style1 is swum  by Swimmer3
Style Style2 is swum  by Swimmer6
Style Style3 is swum  by Swimmer16
Style Style4 is swum  by Swimmer1
Style Style5 is swum  by Swimmer14
Style Style6 is swum  by Swimmer7
Style Style7 is swum  by Swimmer13
Style Style8 is swum  by Swimmer19
Style Style9 is swum  by Swimmer5
Style Style10 is swum  by Swimmer10
Style Style11 is swum  by Swimmer12
Style Style12 is swum  by Swimmer9
Style Style13 is swum  by Swimmer4
Style Style14 is swum  by Swimmer11
Style Style15 is swum  by Swimmer8
Style Style16 is swum  by Swimmer17
Style Style17 is swum  by Swimmer2
Style Style18 is swum  by Swimmer18
Style Style19 is swum  by Swimmer15
Style Style20 is swum  by Swimmer20

real    0m0.028s
user    0m0.026s
sys     0m0.001s
# channeling model:
```

```
Total time: 903
Swimmer Swimmer1 is swims Style3
Swimmer Swimmer2 is swims Style6
Swimmer Swimmer3 is swims Style1
Swimmer Swimmer4 is swims Style8
Swimmer Swimmer5 is swims Style17
Swimmer Swimmer6 is swims Style2
Swimmer Swimmer7 is swims Style13
Swimmer Swimmer8 is swims Style4
Swimmer Swimmer9 is swims Style12
Swimmer Swimmer10 is swims Style10
Swimmer Swimmer11 is swims Style11
Swimmer Swimmer12 is swims Style9
Swimmer Swimmer13 is swims Style7
Swimmer Swimmer14 is swims Style16
Swimmer Swimmer15 is swims Style19
Swimmer Swimmer16 is swims Style14
Swimmer Swimmer17 is swims Style5
Swimmer Swimmer18 is swims Style18
Swimmer Swimmer19 is swims Style15
Swimmer Swimmer20 is swims Style20
or in the dual view
Style Style1 is swum  by Swimmer3
Style Style2 is swum  by Swimmer6
Style Style3 is swum  by Swimmer1
Style Style4 is swum  by Swimmer8
Style Style5 is swum  by Swimmer17
Style Style6 is swum  by Swimmer2
Style Style7 is swum  by Swimmer13
Style Style8 is swum  by Swimmer4
Style Style9 is swum  by Swimmer12
Style Style10 is swum  by Swimmer10
Style Style11 is swum  by Swimmer11
Style Style12 is swum  by Swimmer9
Style Style13 is swum  by Swimmer7
Style Style14 is swum  by Swimmer16
Style Style15 is swum  by Swimmer19
Style Style16 is swum  by Swimmer14
Style Style17 is swum  by Swimmer5
Style Style18 is swum  by Swimmer18
Style Style19 is swum  by Swimmer15
Style Style20 is swum  by Swimmer20
real    0m8.826s
user    0m8.792s
sys     0m0.020s
```

In [12]:
```
!cat exercises/swimmers/.solution/swimmers.pi
```

```
import cp.

main => main(["primal"]).

main([Model]) => main([Model, "sample"]).

main([Model, Instance]) =>

    if Instance = "sample" then
        cl(sample_instance),
        sample_instance(SwimmerNames, StyleNames, Times)
    elseif Instance = "random" then
        cl(random_instance),
        MinTime = 40,
        MaxTime = 80,
        N = 20,
        K = 20,
        random_instance(N, K, MinTime, MaxTime, SwimmerNames, StyleNames, Ti
mes)
    else
        println("Unknown instance type")
    end,
    go(SwimmerNames, StyleNames, Times, Model).

go(SwimmerNames, StyleNames, Times, Model) =>

    printf("# %w model:\n", Model),

    if Model = "primal" then
        primal_model(StyleOfSwimmer, Times, TotalTime),
        solve([$min(TotalTime), StyleOfSwimmer]),
        printf("Total time: %d\n", TotalTime),
        foreach(I in 1..SwimmerNames.length)
            printf("Swimmer %w is swims %w\n", SwimmerNames[I], StyleNames[S
tyleOfSwimmer[I]])
        end
    elseif Model = "dual" then
        dual_model(SwimmerOfStyle, Times, TotalTime),
        solve([$min(TotalTime), SwimmerOfStyle]),
        printf("Total time: %d\n", TotalTime),
        foreach(J in 1..StyleNames.length)
            printf("Style %w is swum  by %w\n", StyleNames[J], SwimmerNames
[SwimmerOfStyle[J]])
        end
    elseif Model = "channeling" then
        channeling(StyleOfSwimmer, SwimmerOfStyle, Times, TotalTime),
        solve([$min(TotalTime), StyleOfSwimmer ++ SwimmerOfStyle]),
        printf("Total time: %d\n", TotalTime),
        foreach(I in 1..SwimmerNames.length)
            printf("Swimmer %w is swims %w\n", SwimmerNames[I], StyleNames[S
tyleOfSwimmer[I]])
        end,
        println("or in the dual view"),
        foreach(J in 1..StyleNames.length)
            printf("Style %w is swum  by %w\n", StyleNames[J], SwimmerNames
[SwimmerOfStyle[J]])
```

```
        end
    else
        printf("Unknown model: %w\n", Model)
    end.


primal_model(StyleOfSwimmer, Times, TotalTime) =>
    N = Times.length,
    %K = Times[1].length,
    K = N,
    StyleOfSwimmer = new_array(N),
    StyleOfSwimmer :: 1..K,
    all_different(StyleOfSwimmer),

    TimeOfSwimmer = new_array(N),
    foreach(I in 1..N)
        matrix_element(Times, I, StyleOfSwimmer[I], TimeOfSwimmer[I])
    end,
    TotalTime #= sum(TimeOfSwimmer).

dual_model(SwimmerOfStyle, Times, TotalTime) =>
    N = Times.length,
    K = N,
    SwimmerOfStyle = new_array(K),
    SwimmerOfStyle :: 1..N,
    all_different(SwimmerOfStyle),

    TimeOfStyle = new_array(K),
    foreach(J in 1..K)
        matrix_element(Times, J, SwimmerOfStyle[J], TimeOfStyle[J])
    end,
    TotalTime #= sum(TimeOfStyle).

channeling(StyleOfSwimmer, SwimmerOfStyle, Times, TotalTime) =>
    primal_model(StyleOfSwimmer, Times, TotalTime),
    dual_model(SwimmerOfStyle, Times, TotalTime),
    assignment(StyleOfSwimmer, SwimmerOfStyle).  %chanelling constraint
```

Modelling functions

In general, how to model a function (mapping) $f : A \to B$? Let's say $A = \{1, \ldots, n\}$ and $B = \{1, \ldots, k\}$.

- as an array:

  ```
  F = new_array(N},
  F :: 1..K.
  ```

- *injective*: `all_different(F)`
- *surjective*: a partition of A into classes labelled by B, to each element of B map a set of elements of A. In Picat we can model set as their characteristic

vectors. More on modelling with sets later.

- *partial function*: a dummy value for undefined inputs
- *dual model*: switch the role of variables and values (not a function unless F injective, see above)
- *channelling*: combine the primal and dual models, if it is a bijection, then use `assignment(F, FInv)`

In [13]:
```
!picat functions.pi 4 4
```

```
{1,2,3,4}
{1,2,3,4}
```

In [14]:
```
!cat functions.pi
```

```
import cp.

main([N, K]) =>
    N := N.to_int,
    K := K.to_int,

    % function
    F = new_array(N),
    F :: 1..K,

    % injective
    all_different(F),

    % dual model if it is a bijection (K=N and injective)
    FInv = new_array(K),
    FInv :: 1..N,

    % channeling if it is a bijection (K=N and injective)
    assignment(F, FInv),

    % % dual model in general
    % FInv = new_array(K, N),
    % FInv :: 0..1,

    % % surjective in general
    % foreach(J in 1..N)
    %     sum([FInv[I, J]: I in 1..K]) #>= 1
    % end,

    % % channeling in general
    % foreach(I in 1..N)
    %     foreach(J in 1..N)
    %         (FInv[J, I] #= 1) #<=> (F[I] #= J)
    %     end
    % end,

    solve(F ++ FInv),
    println(F),
    println(FInv).
```