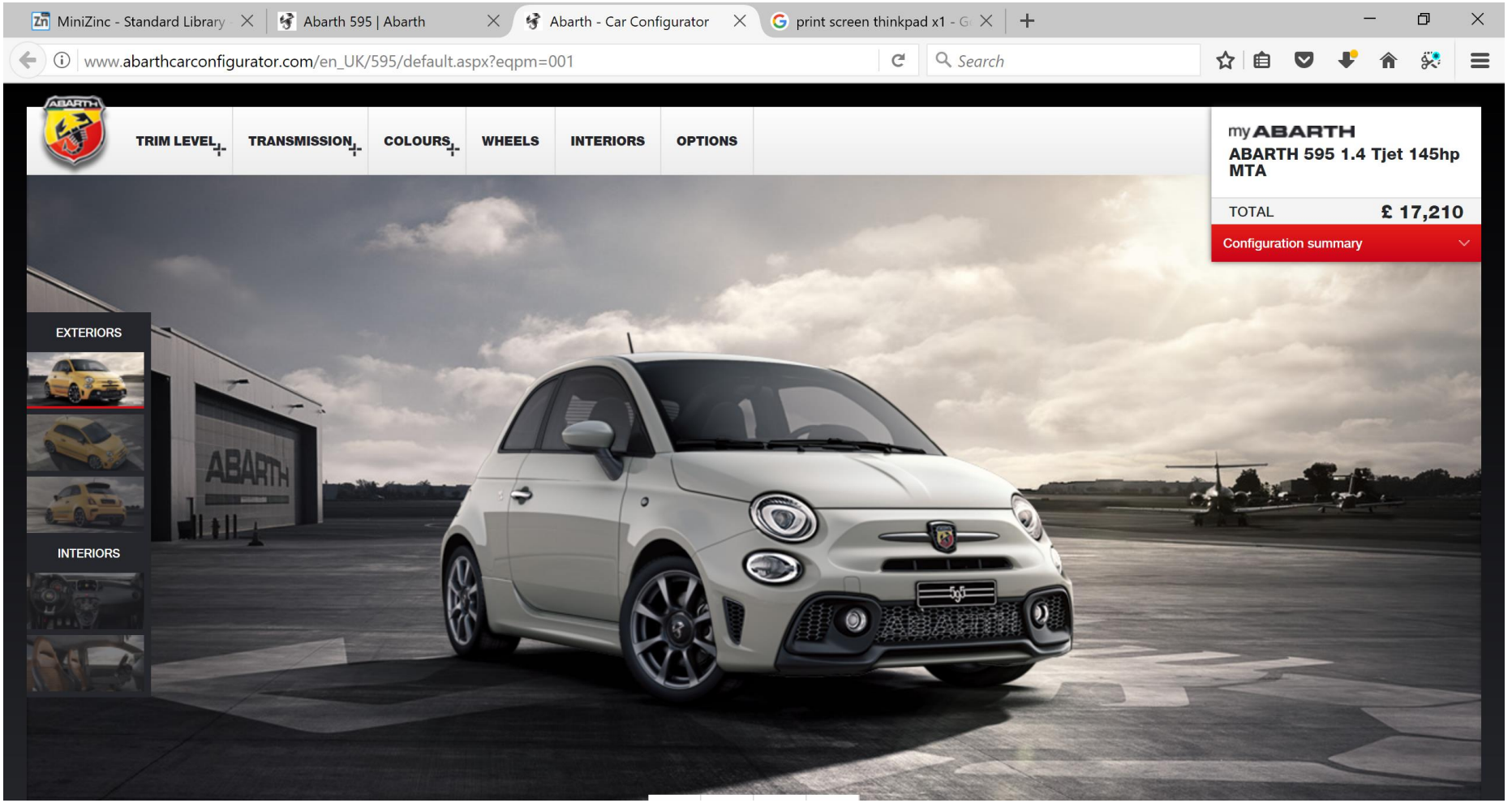
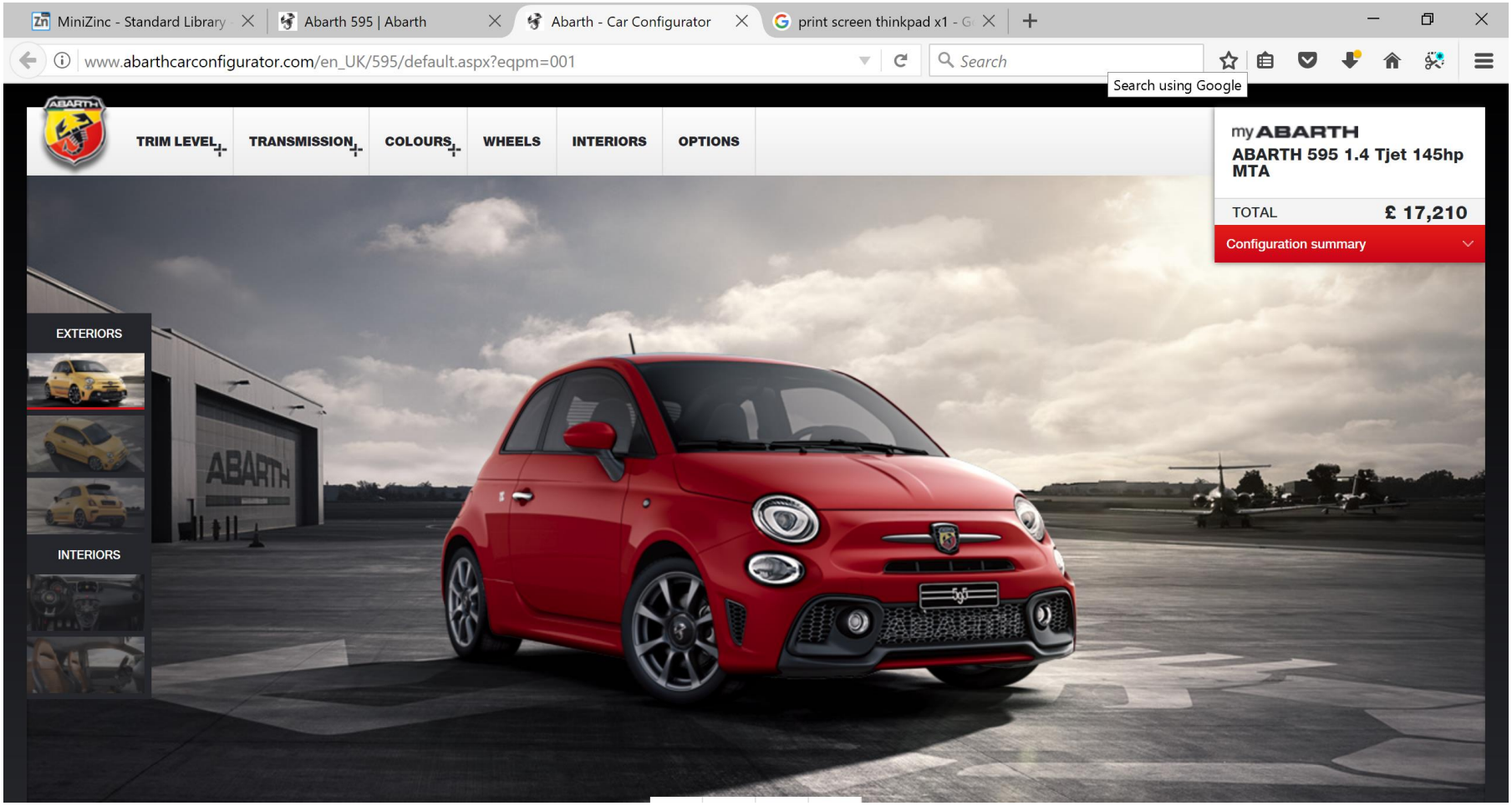
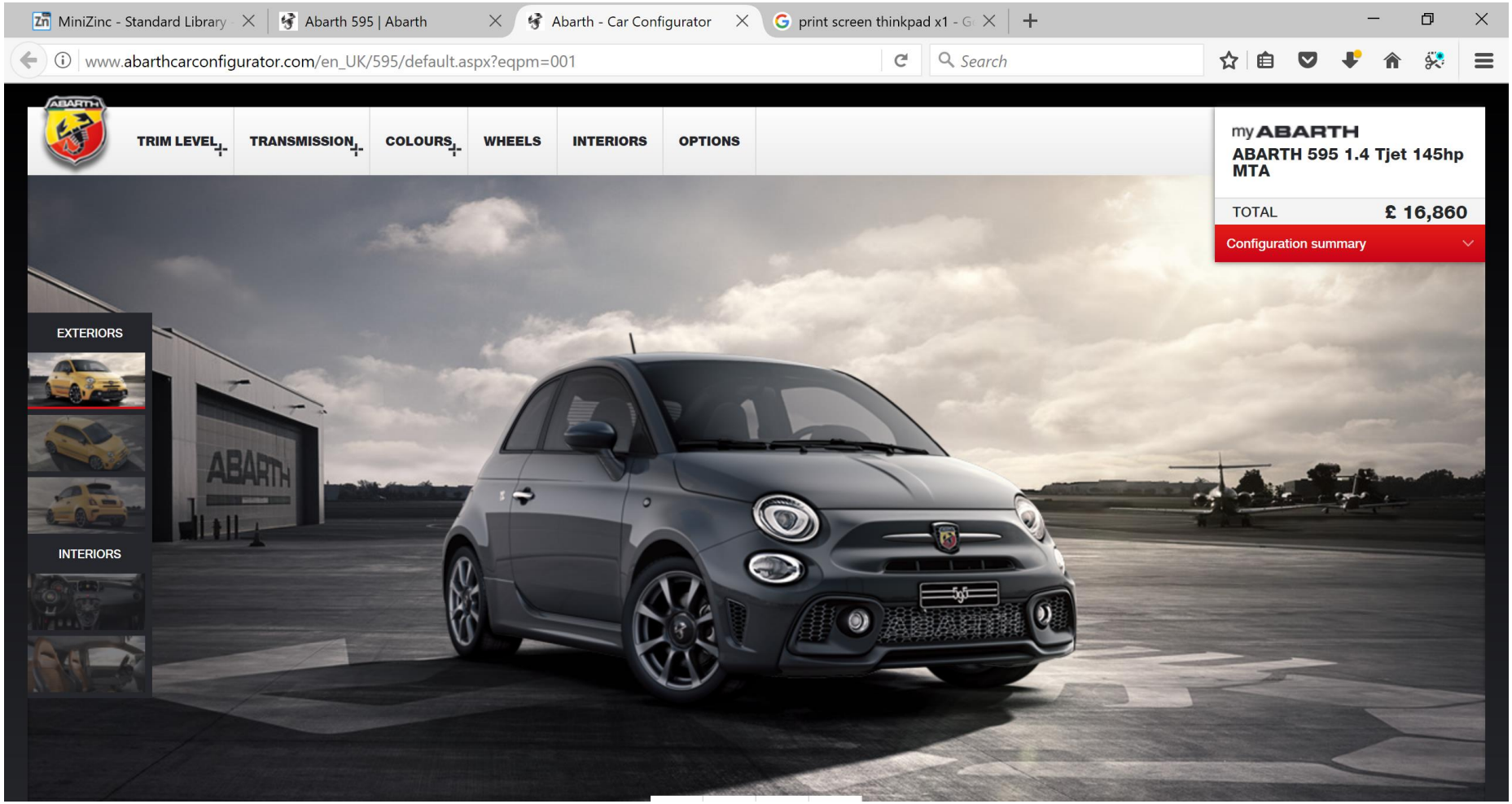
Abarth Paint Shop

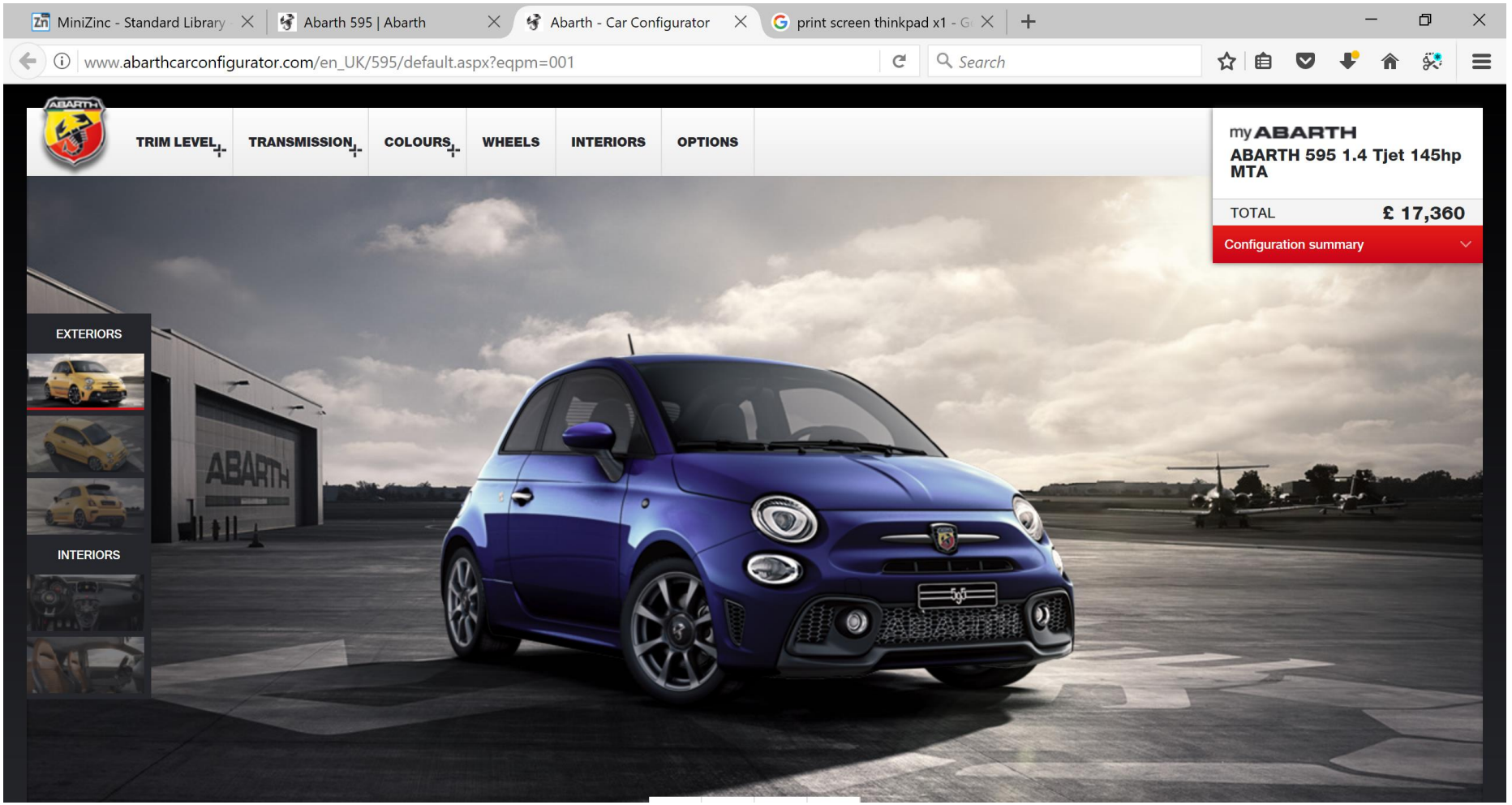
In the Abarth paint shop cars can be :

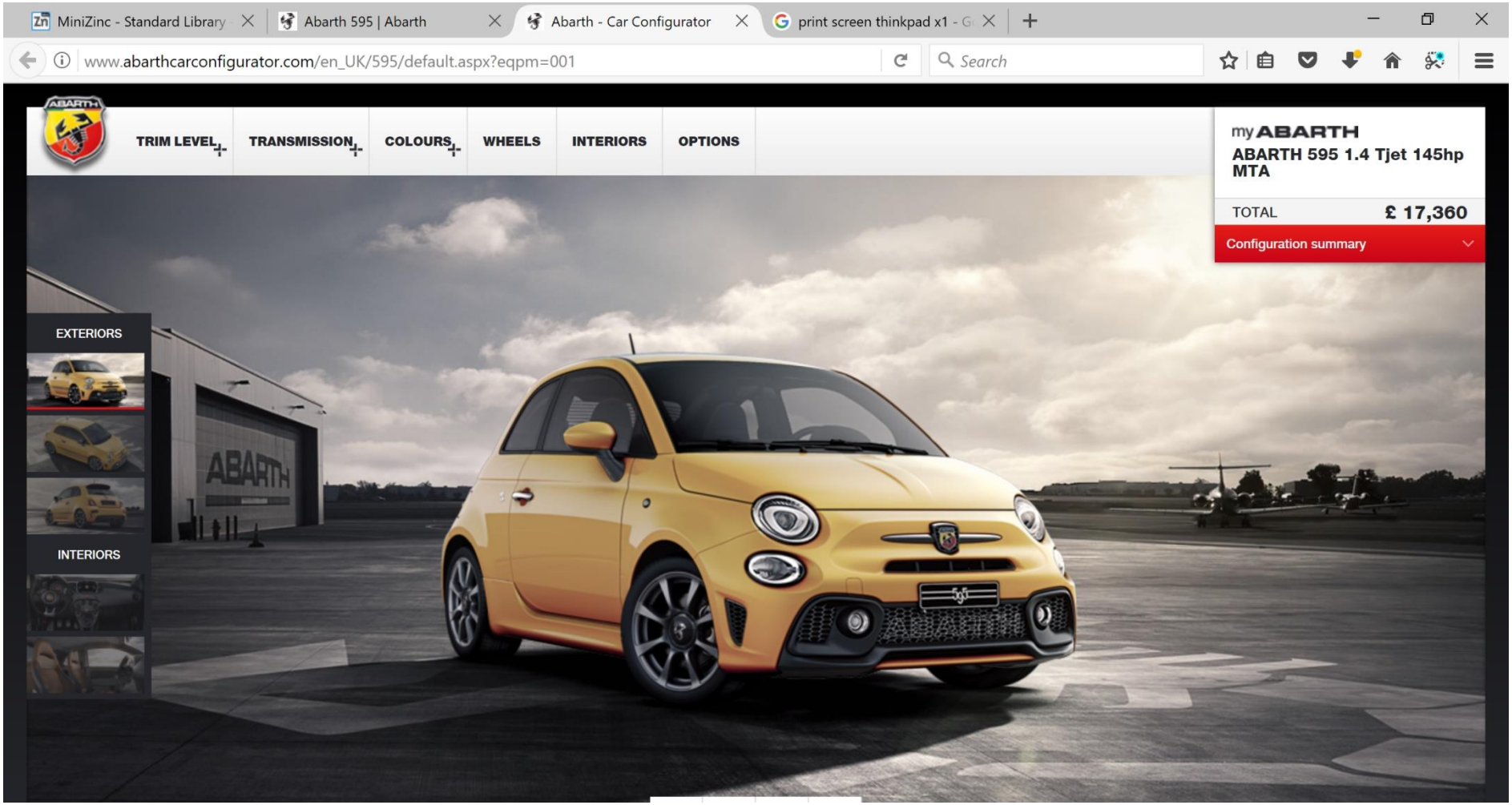
grey
red
white
yellow
blue … or
black

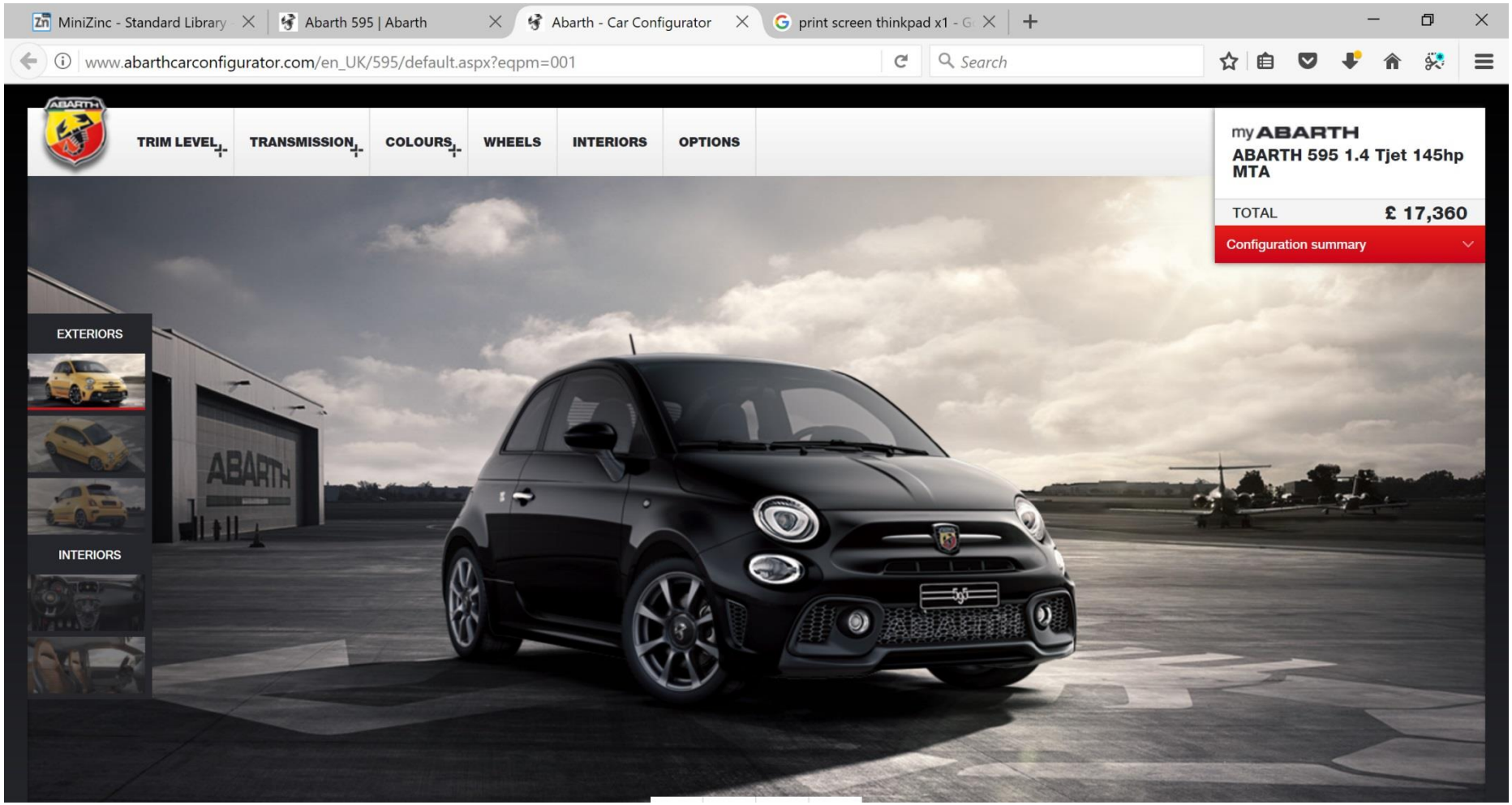

MiniZinc - Standard Library
Abarth 595 | Abarth
Abarth - Car Configurator
print screen thinkpad x1 - G
www.abarthcarconfigurator.com/en_UK/595/default.aspx?eqpm=001
Search
TRIM LEVEL
TRANSMISSION
COLOURS
WHEELS
INTERIORS
OPTIONS
my ABARTH
ABARTH 595 1.4 Tjet 145hp MTA
TOTAL
£ 17,210
Configuration summary
EXTERIORS
INTERIORS
ABARTH

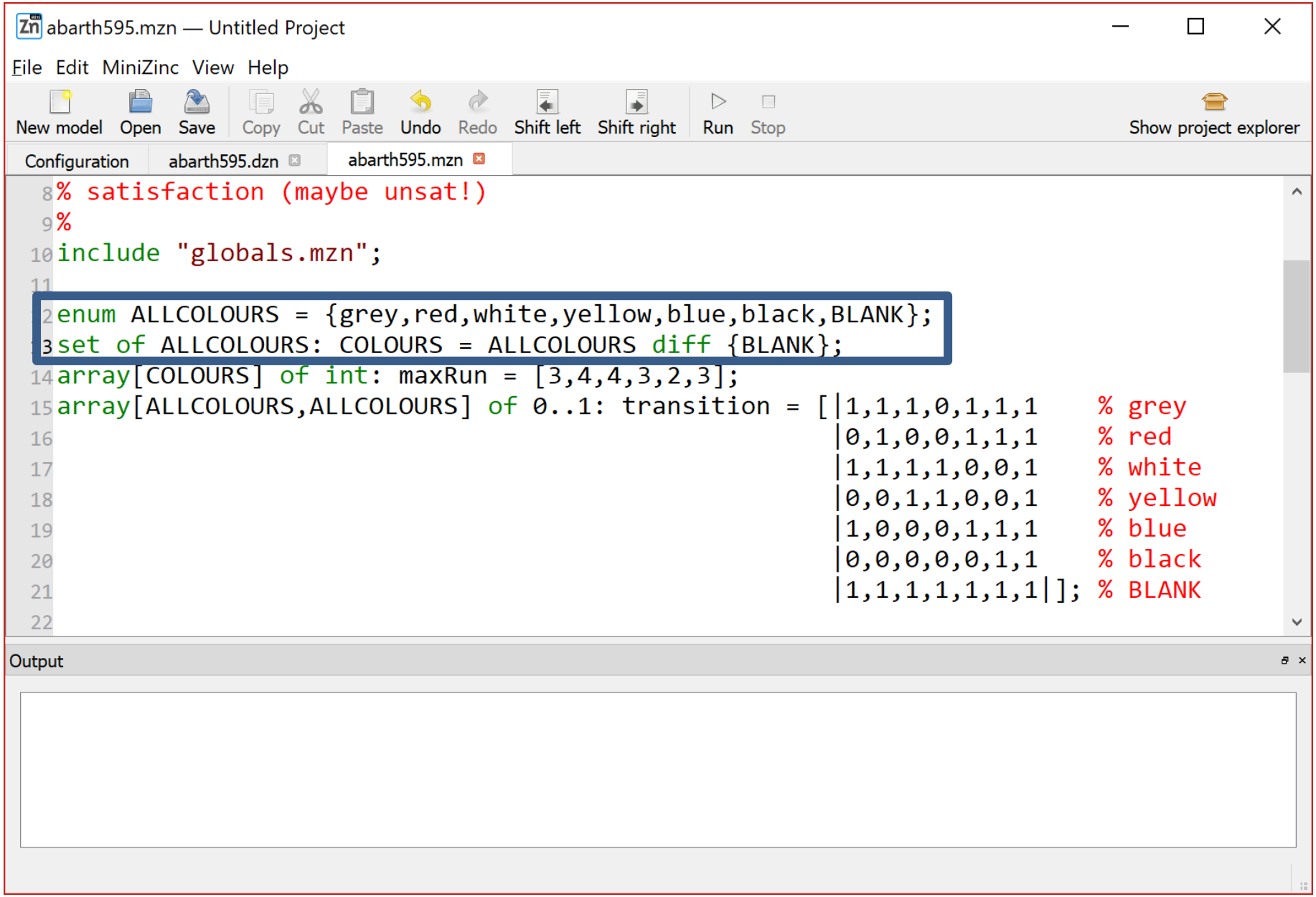

MiniZinc - Standard Library
Abarth 595 | Abarth
Abarth - Car Configurator
print screen thinkpad x1 - G
www.abarthcarconfigurator.com/en_UK/595/default.aspx?eqpm=001
Search
Search using Google
TRIM LEVEL
TRANSMISSION
COLOURS
WHEELS
INTERIORS
OPTIONS
my ABARTH
ABARTH 595 1.4 Tjet 145hp MTA
TOTAL
£ 17,210
Configuration summary
EXTERIORS
INTERIORS
ABARTH


MiniZinc - Standard Library
Abarth 595 | Abarth
Abarth - Car Configurator
print screen thinkpad x1 - G
www.abarthcarconfigurator.com/en_UK/595/default.aspx?eqpm=001
Search
TRIM LEVEL
TRANSMISSION
COLOURS
WHEELS
INTERIORS
OPTIONS
my ABARTH
ABARTH 595 1.4 Tjet 145hp MTA
TOTAL
£ 16,860
Configuration summary
EXTERIORS
INTERIORS
ABARTH


MiniZinc - Standard Library
Abarth 595 | Abarth
Abarth - Car Configurator
print screen thinkpad x1 - G
www.abarthcarconfigurator.com/en_UK/595/default.aspx?eqpm=001
Search
TRIM LEVEL
TRANSMISSION
COLOURS
WHEELS
INTERIORS
OPTIONS
my ABARTH
ABARTH 595 1.4 Tjet 145hp MTA
TOTAL
£ 17,360
Configuration summary
EXTERIORS
INTERIORS
ABARTH

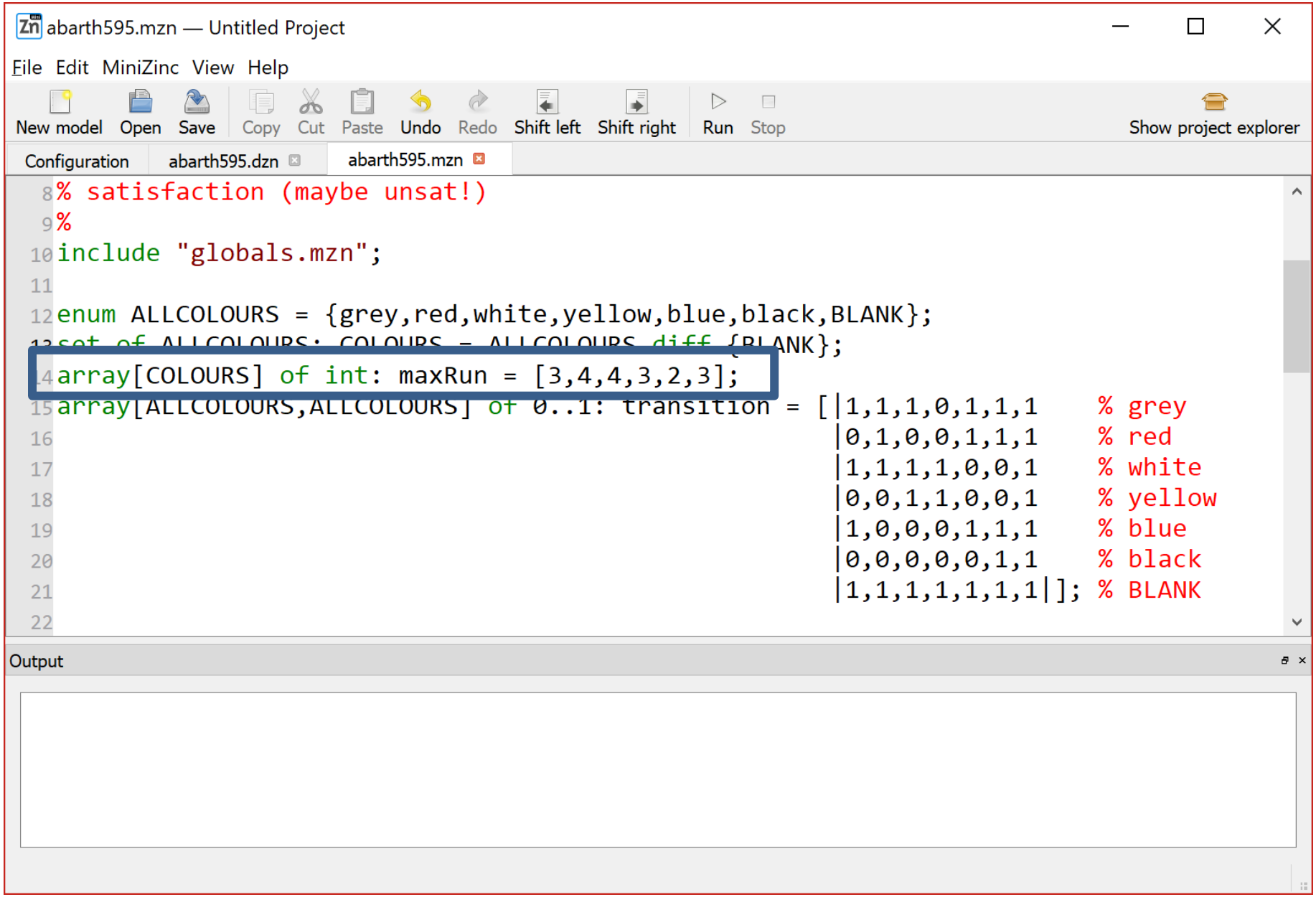

MiniZinc - Standard Library
Abarth 595 | Abarth
Abarth - Car Configurator
print screen thinkpad x1 - G
www.abarthcarconfigurator.com/en_UK/595/default.aspx?eqpm=001
Search
TRIM LEVEL
TRANSMISSION
COLOURS
WHEELS
INTERIORS
OPTIONS
myABARTH
ABARTH 595 1.4 Tjet 145hp MTA
TOTAL
£ 17,360
Configuration summary
EXTERIORS
INTERIORS
ABARTH


MiniZinc - Standard Library
Abarth 595 | Abarth
Abarth - Car Configurator
print screen thinkpad x1 - G
www.abarthcarconfigurator.com/en_UK/595/default.aspx?eqpm=001
Search
TRIM LEVEL
TRANSMISSION
COLOURS
WHEELS
INTERIORS
OPTIONS
my ABARTH
ABARTH 595 1.4 Tjet 145hp MTA
TOTAL
£ 17,360
Configuration summary
EXTERIORS
INTERIORS
ABARTH


abarth595.mzn — Untitled Project
File Edit MiniZinc View Help
New model Open Save Copy Cut Paste Undo Redo Shift left Shift right Run Stop Show project explorer
Configuration abarth595.dzn abarth595.mzn
% satisfaction (maybe unsat!)
%
include "globals.mzn";
enum ALLCOLOURS = {grey,red,white,yellow,blue,black,BLANK};
set of ALLCOLOURS: COLOURS = ALLCOLOURS diff {BLANK};
array[COLOURS] of int: maxRun = [3,4,4,3,2,3];
array[ALLCOLOURS,ALLCOLOURS] of 0..1: transition = [|1,1,1,0,1,1,1 % grey
|0,1,0,0,1,1,1 % red
|1,1,1,1,0,0,1 % white
|0,0,1,1,0,0,1 % yellow
|1,0,0,0,1,1,1 % blue
|0,0,0,0,0,1,1 % black
|1,1,1,1,1,1,1|]; % BLANK
Output

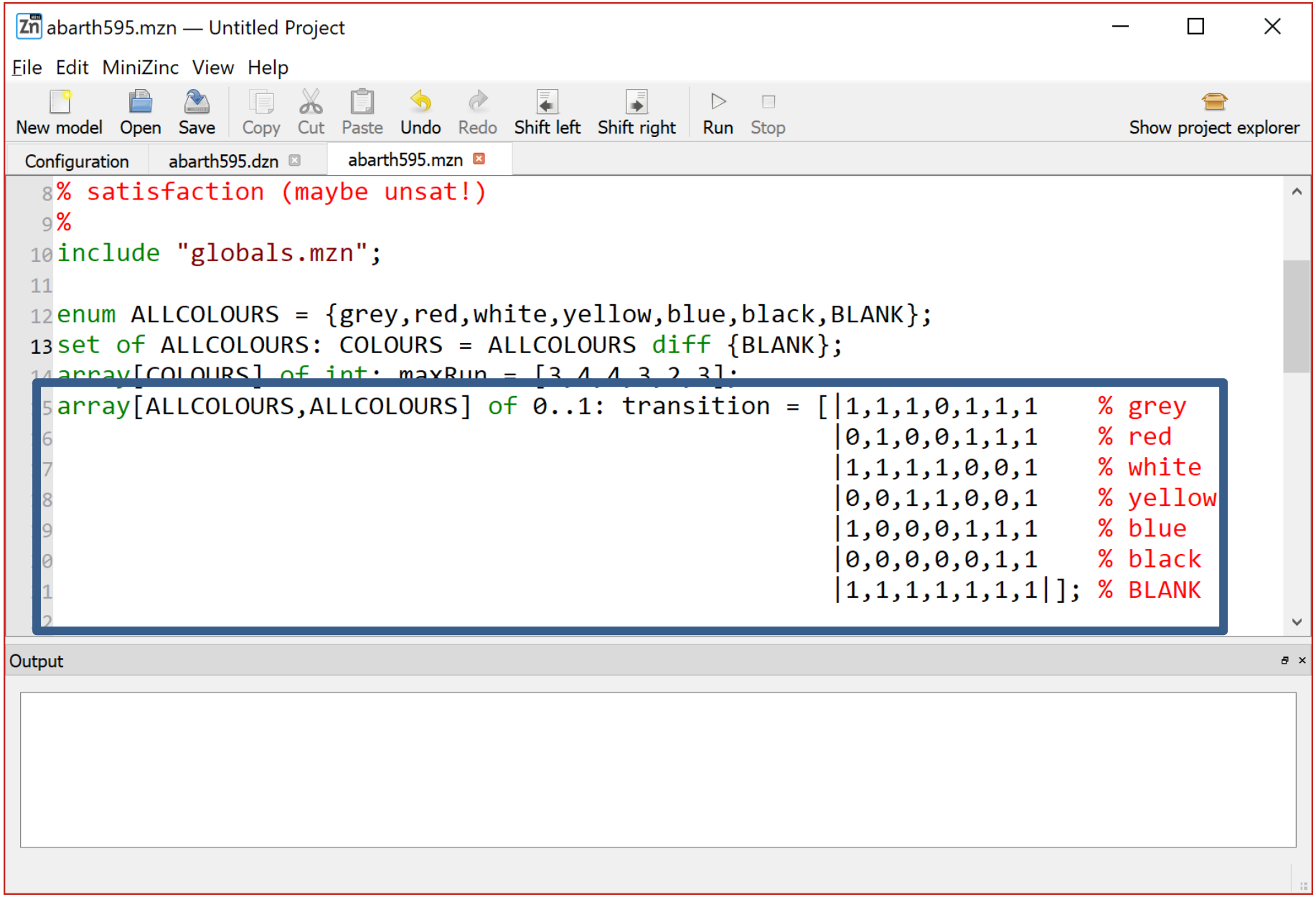
When painting cars …

- 3 grey cars can be painted one after the other
- 4 red can be painted one after the other
- 4 white can be painted one after the other
- 3 yellow one after the other
- 2 blue one after the other
- 3 black one after the other


abarth595.mzn — Untitled Project
File Edit MiniZinc View Help
New model Open Save Copy Cut Paste Undo Redo Shift left Shift right Run Stop Show project explorer
Configuration abarth595.dzn abarth595.mzn
% satisfaction (maybe unsat!)
%
include "globals.mzn";
enum ALLCOLOURS = {grey,red,white,yellow,blue,black,BLANK};
set of ALLCOLOURS: COLOURS = ALLCOLOURS diff {BLANK};
array[COLOURS] of int: maxRun = [3,4,4,3,2,3];
array[ALLCOLOURS,ALLCOLOURS] of 0..1: transition = [|1,1,1,0,1,1,1 % grey
|0,1,0,0,1,1,1 % red
|1,1,1,1,0,0,1 % white
|0,0,1,1,0,0,1 % yellow
|1,0,0,0,1,1,1 % blue
|0,0,0,0,0,1,1 % black
|1,1,1,1,1,1,1|]; % BLANK
Output

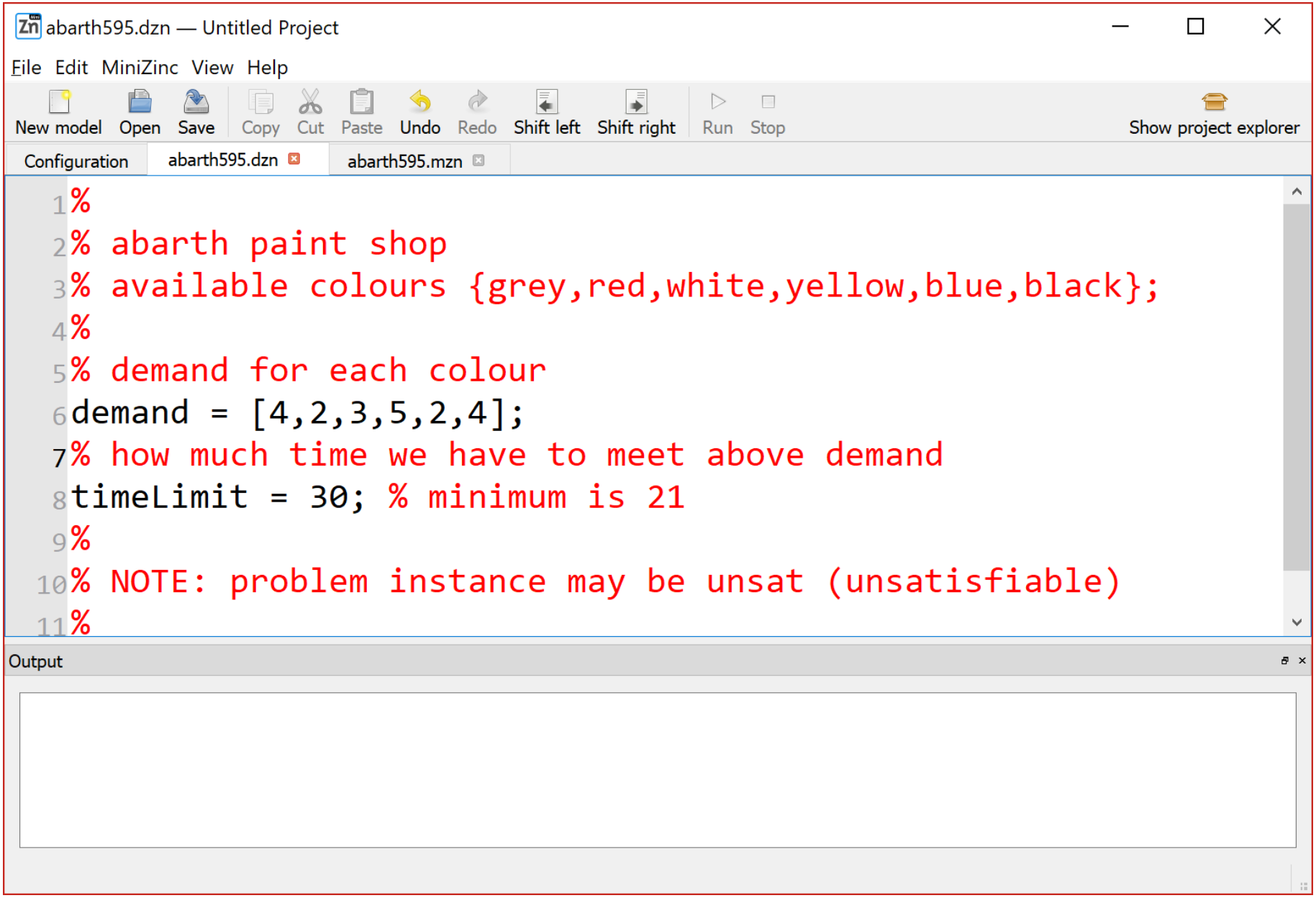
When changing colours …

- After grey we can paint grey, red, white, blue, black or BLANK
- After red we can paint red, blue, black or BLANK
- After white we can paint white, grey, red, yellow or BLANK
- After yellow we can paint yellow, white or BLANK
- After blue we can paint blue, black or BLANK
- After black we can paint black or BLANK
- After BLANK we have clean paint guns and can paint any colour


abarth595.mzn — Untitled Project
File Edit MiniZinc View Help
New model Open Save Copy Cut Paste Undo Redo Shift left Shift right Run Stop Show project explorer
Configuration abarth595.dzn abarth595.mzn
% satisfaction (maybe unsat!)
%
include "globals.mzn";
enum ALLCOLOURS = {grey,red,white,yellow,blue,black,BLANK};
set of ALLCOLOURS: COLOURS = ALLCOLOURS diff {BLANK};
array[ALLCOLOURS,ALLCOLOURS] of 0..1: transition = [|1,1,1,0,1,1,1 % grey
|0,1,0,0,1,1,1 % red
|1,1,1,1,0,0,1 % white
|0,0,1,1,0,0,1 % yellow
|1,0,0,0,1,1,1 % blue
|0,0,0,0,0,1,1 % black
|1,1,1,1,1,1,1|]; % BLANK
Output

Demand …

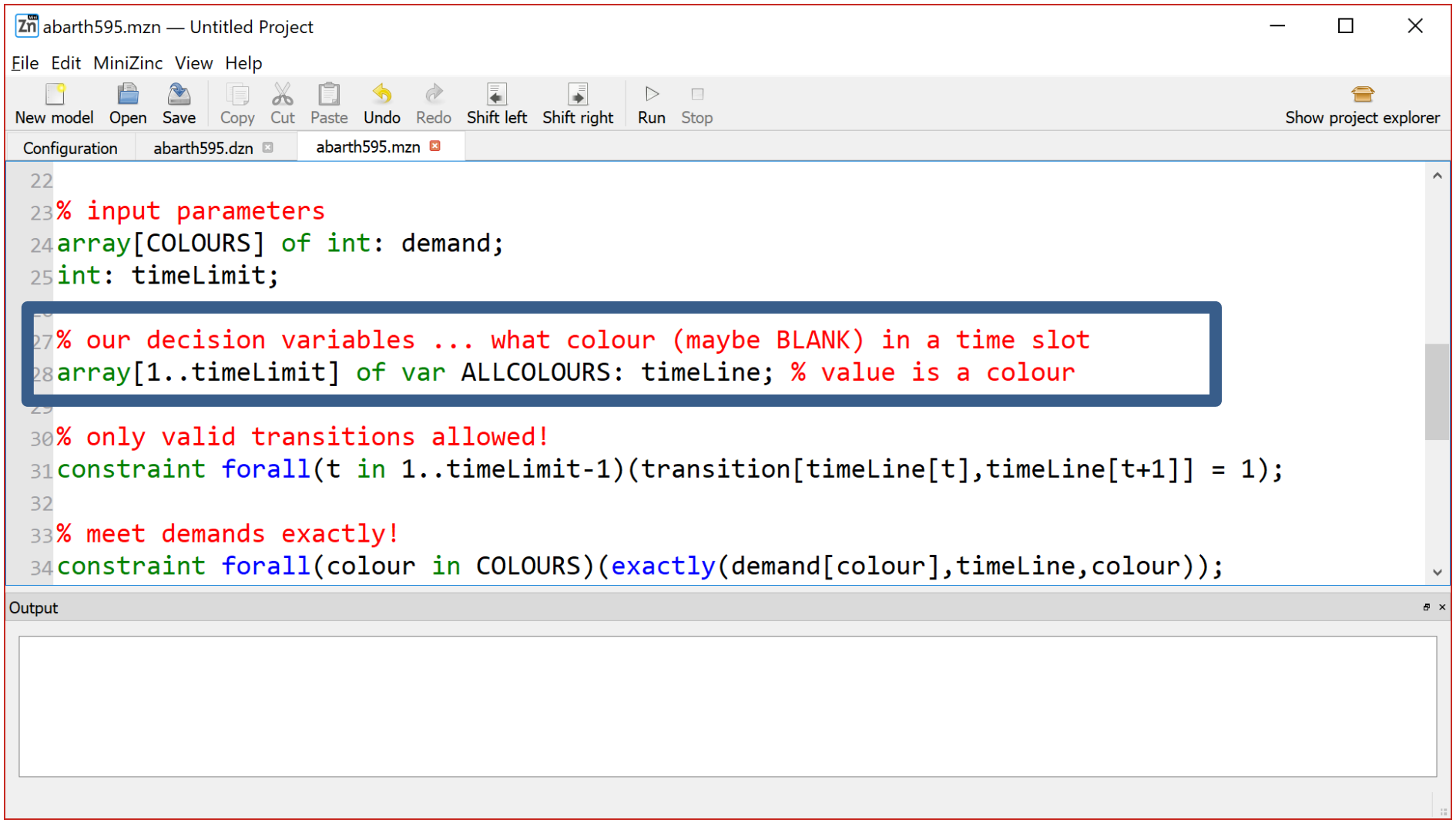
We have a demand for cars of these colours
and an allowed time to paint them …


abarth595.dzn — Untitled Project
File Edit MiniZinc View Help
New model Open Save Copy Cut Paste Undo Redo Shift left Shift right Run Stop Show project explorer
Configuration abarth595.dzn abarth595.mzn
1 %
2 % abarth paint shop
3 % available colours {grey,red,white,yellow,blue,black};
4 %
5 % demand for each colour
6 demand = [4,2,3,5,2,4];
7 % how much time we have to meet above demand
8 timeLimit = 30; % minimum is 21
9 %
10 % NOTE: problem instance may be unsat (unsatisfiable)
11 %
Output

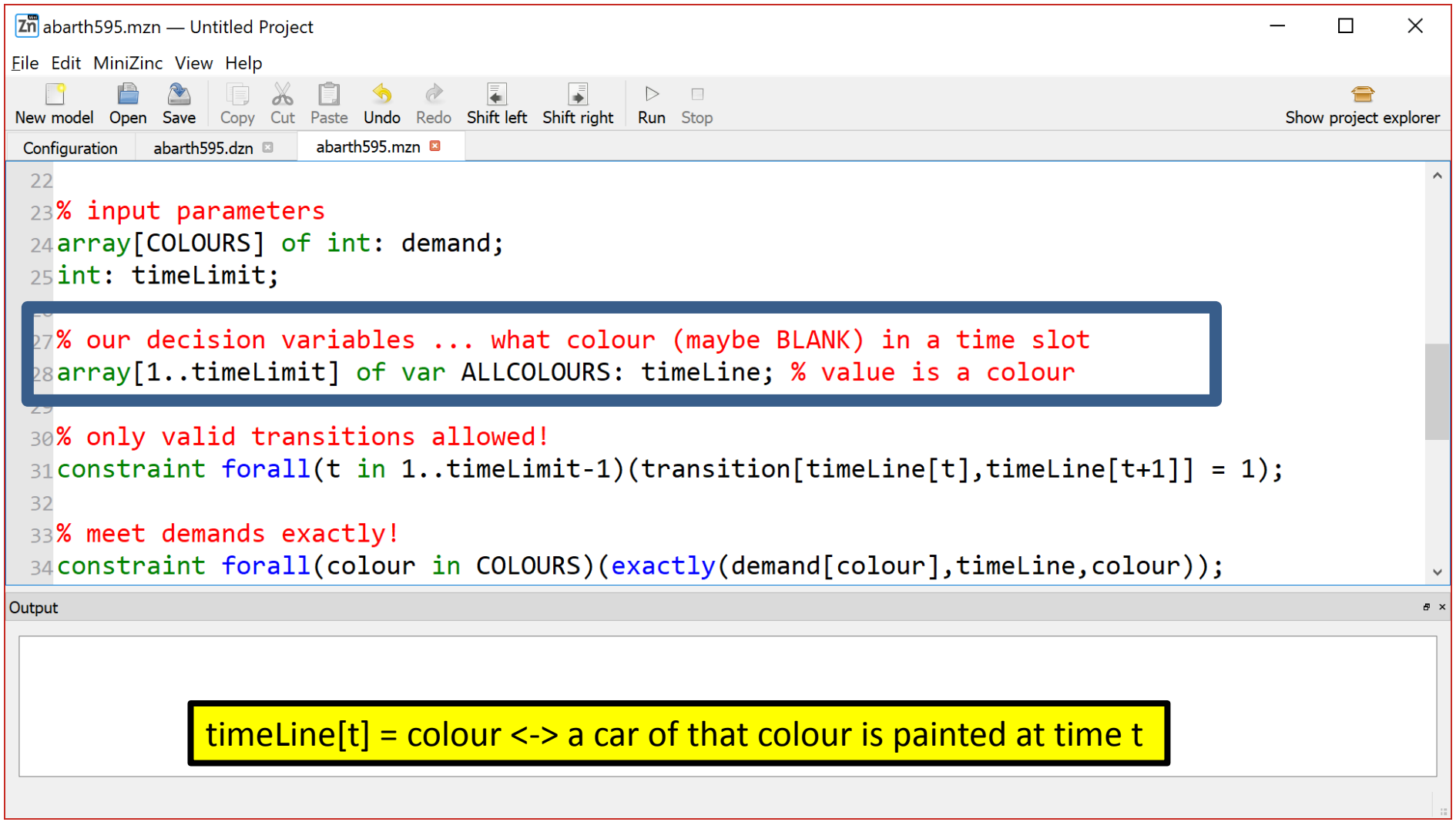
Decision variables …

Given a time line, at each time slot, what colour of car will we paint?

Decision variables ...



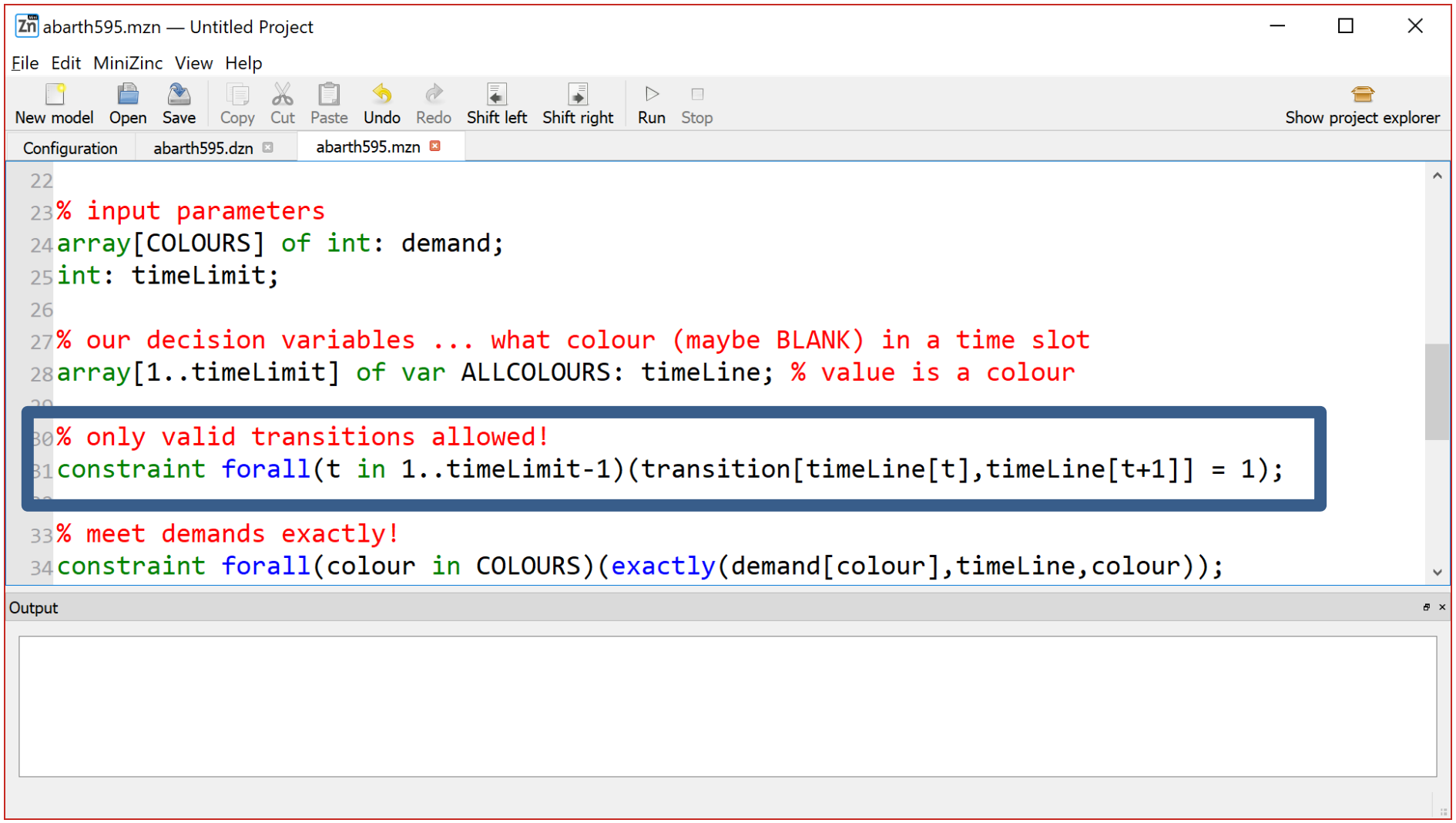
Decision variables ...



Colour transitions must be valid …

When we change colours , transitions, they must be valid

Colour transitions must be valid …



Colour transitions must be valid …

abarth595.mzn — Untitled Project

File Edit MiniZinc View Help

New model | Open | Save | Copy | Cut | Paste | Undo | Redo | Shift left | Shift right | Run | Stop | Show project explorer

Configuration | abarth595.dzn | abarth595.mzn

```
22
23 % input parameters
24 array[COLOURS] of int: demand;
25 int: timeLimit;
26
27 % our decision variables ... what colour (maybe BLANK) in a time slot
28 array[1..timeLimit] of var ALLCOLOURS: timeLine; % value is a colour
29
30 % only valid transitions allowed!
31 constraint forall(t in 1..timeLimit-1)(transition[timeLine[t],timeLine[t+1]] = 1);
32
33 % meet demands exactly!
34 constraint forall(colour in COLOURS)(exactly(demand[colour],timeLine,colour));
```

Output

Using constrained integer variable as array index!

Colour transitions must be valid …

abarth595.mzn — Untitled Project

File Edit MiniZinc View Help

New model | Open | Save | Copy | Cut | Paste | Undo | Redo | Shift left | Shift right | Run | Stop | Show project explorer

Configuration | abarth595.dzn | abarth595.mzn

```
22
23 % input parameters
24 array[COLOURS] of int: demand;
25 int: timeLimit;
26
27 % our decision variables ... what colour (maybe BLANK) in a time slot
28 array[1..timeLimit] of var ALLCOLOURS: timeLine; % value is a colour
29
30 % only valid transitions allowed!
31 constraint forall(t in 1..timeLimit-1)(transition[timeLine[t],timeLine[t+1]] = 1);
32
33 % meet demands exactly!
34 constraint forall(colour in COLOURS)(exactly(demand[colour],timeLine,colour));
```

Output

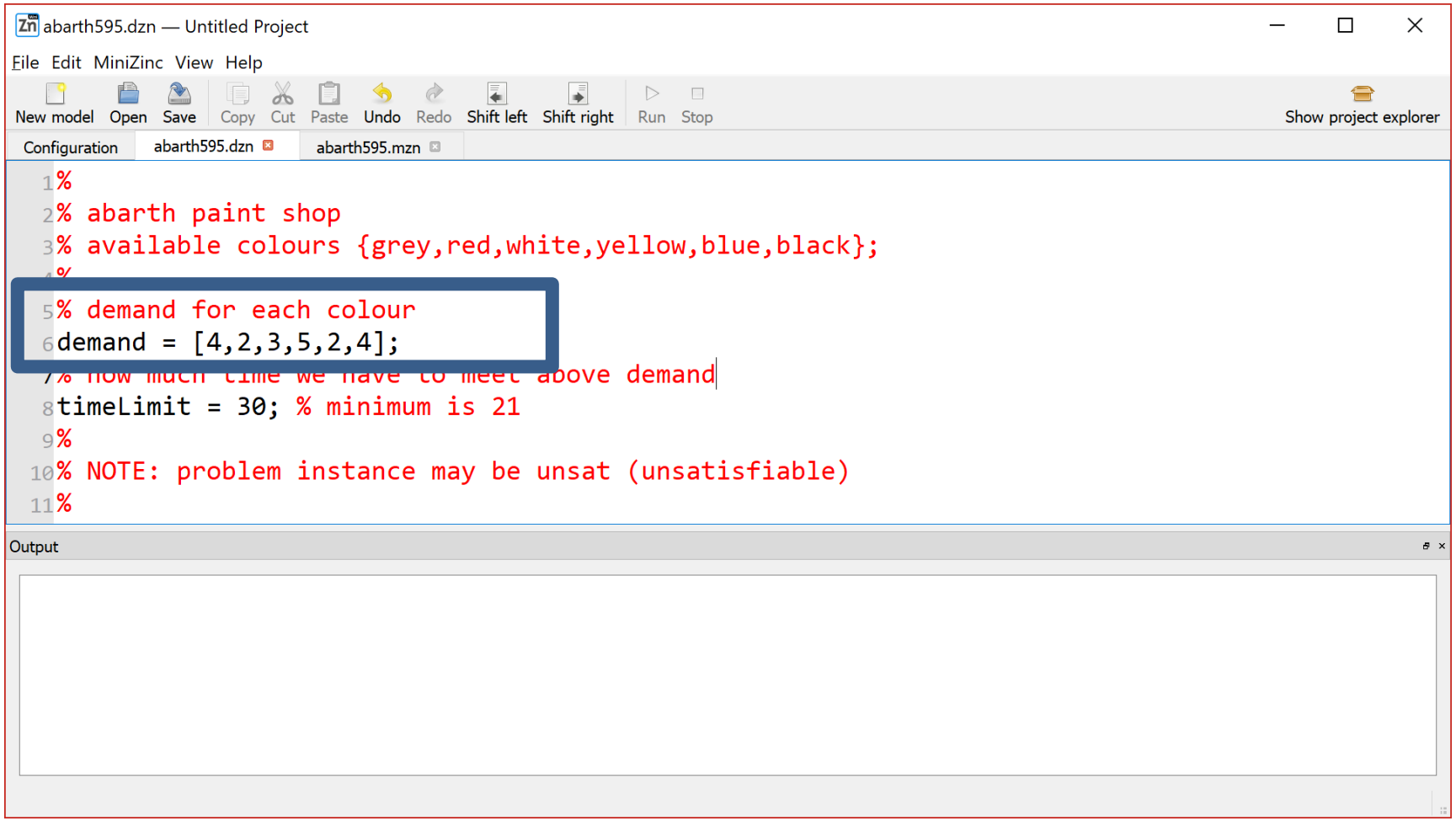
These are colours!

Using constrained integer variable as array index!

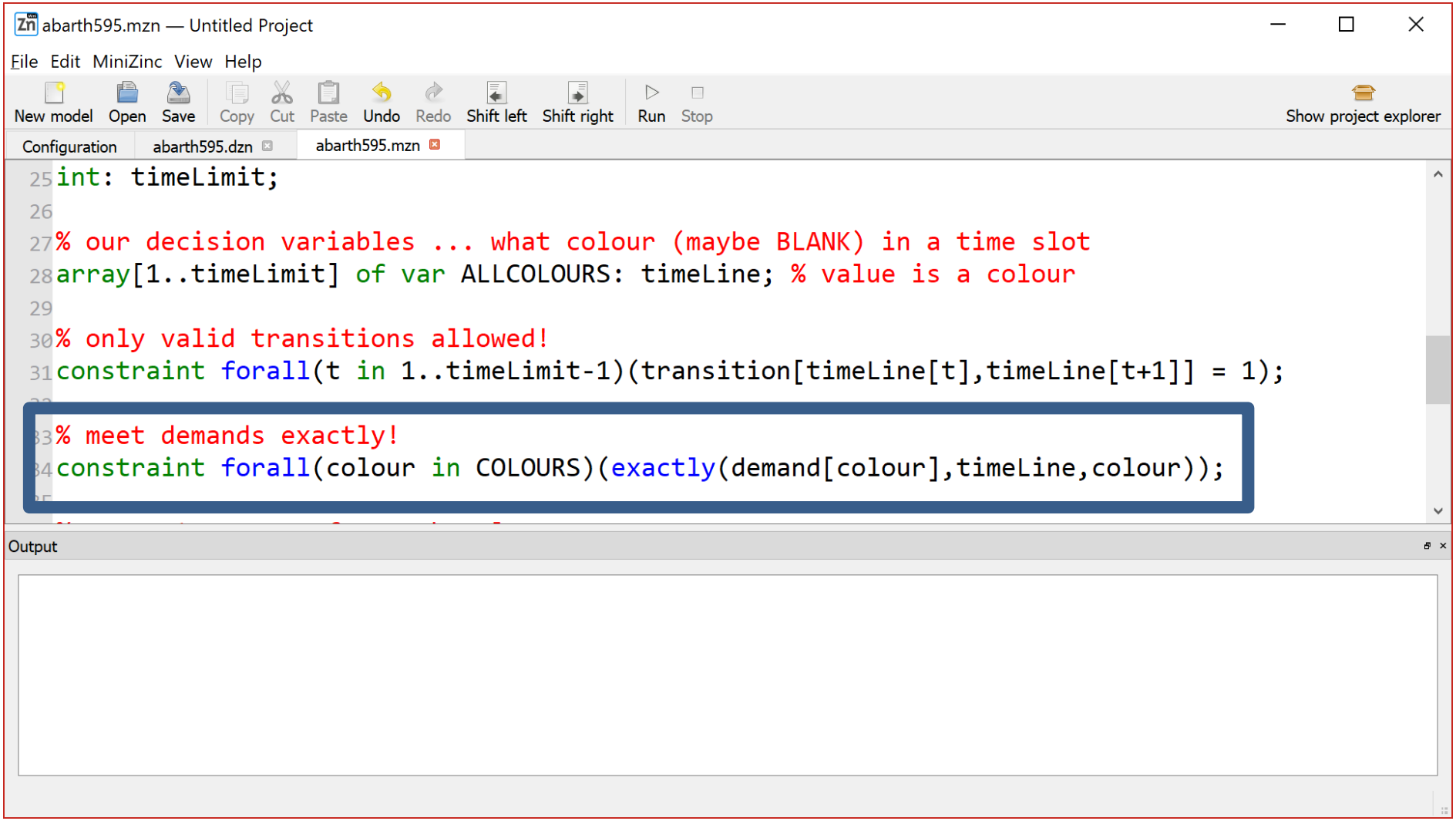
Demand for each colour must be met, exactly !!!!

Meet demand exactly …

Demand for each colour must be met, exactly !!!!



Demand for each colour must be met, exactly !!!!



Demand for each colour must be met, exactly !!!!

abarth595.mzn — Untitled Project

File Edit MiniZinc View Help

New model Open Save Copy Cut Paste Undo Redo Shift left Shift right Run Stop Show project explorer

Configuration abarth595.dzn abarth595.mzn

```
25 int: timeLimit;
26
27 % our decision variables ... what colour (maybe BLANK) in a time slot
28 array[1..timeLimit] of var ALLCOLOURS: timeLine; % value is a colour
29
30 % only valid transitions allowed!
31 constraint forall(t in 1..timeLimit-1)(transition[timeLine[t],timeLine[t+1]] = 1);
2
3 % meet demands exactly!
4 constraint forall(colour in COLOURS)(exactly(demand[colour],timeLine,colour));
```

Output

NOTE: no demand for BLANK ☺

The maximum run of each colour …

```
enum ALLCOLOURS = {grey,red,white,yellow,blue,black,BLANK};
set of ALLCOLOURS: COLOURS = ALLCOLOURS diff {BLANK};
array[COLOURS] of int: maxRun = [3,4,4,3,2,3];
```

The maximum run of each colour …

```
enum ALLCOLOURS = {grey,red,white,yellow,blue,black,BLANK};
set of ALLCOLOURS: COLOURS = ALLCOLOURS diff {BLANK};
array[COLOURS] of int: maxRun = [3,4,4,3,2,3];
```

The maximum run of each colour …

```
enum ALLCOLOURS = {grey,red,white,yellow,blue,black,BLANK};
set of ALLCOLOURS: COLOURS = ALLCOLOURS diff {BLANK};
array[COLOURS] of int: maxRun = [3,4,4,3,2,3];
```

The maximum run of each colour …

```
enum ALLCOLOURS = {grey,red,white,yellow,blue,black,BLANK};
set of ALLCOLOURS: COLOURS = ALLCOLOURS diff {BLANK};
array[COLOURS] of int: maxRun = [3,4,4,3,2,3];
```

The maximum run of each colour …

```
enum ALLCOLOURS = {grey,red,white,yellow,blue,black,BLANK};
set of ALLCOLOURS: COLOURS = ALLCOLOURS diff {BLANK};
array[COLOURS] of int: maxRun = [3,4,4,3,2,3];
```

The maximum run of each colour …

```
enum ALLCOLOURS = {grey,red,white,yellow,blue,black,BLANK};
set of ALLCOLOURS: COLOURS = ALLCOLOURS diff {BLANK};
array[COLOURS] of int: maxRun = [3,4,4,3,2,3];
```

The maximum run of each colour …

```
enum ALLCOLOURS = {grey,red,white,yellow,blue,black,BLANK};
set of ALLCOLOURS: COLOURS = ALLCOLOURS diff {BLANK};
array[COLOURS] of int: maxRun = [3,4,4,3,2,3];
```

The maximum run of each colour …

```
enum ALLCOLOURS = {grey,red,white,yellow,blue,black,BLANK};
set of ALLCOLOURS: COLOURS = ALLCOLOURS diff {BLANK};
array[COLOURS] of int: maxRun = [3,4,4,3,2,3];
```

The maximum run of each colour …

```
enum ALLCOLOURS = {grey,red,white,yellow,blue,black,BLANK};
set of ALLCOLOURS: COLOURS = ALLCOLOURS diff {BLANK};
array[COLOURS] of int: maxRun = [3,4,4,3,2,3];
```

The maximum run of each colour …

```
enum ALLCOLOURS = {grey,red,white,yellow,blue,black,BLANK};
set of ALLCOLOURS: COLOURS = ALLCOLOURS diff {BLANK};
array[COLOURS] of int: maxRun = [3,4,4,3,2,3];
```

The maximum run of each colour …

```
enum ALLCOLOURS = {grey,red,white,yellow,blue,black,BLANK};
set of ALLCOLOURS: COLOURS = ALLCOLOURS diff {BLANK};
array[COLOURS] of int: maxRun = [3,4,4,3,2,3];
```

The maximum run of each colour …

```
enum ALLCOLOURS = {grey,red,white,yellow,blue,black,BLANK};
set of ALLCOLOURS: COLOURS = ALLCOLOURS diff {BLANK};
array[COLOURS] of int: maxRun = [3,4,4,3,2,3];
```

The maximum run of each colour …

```
enum ALLCOLOURS = {grey,red,white,yellow,blue,black,BLANK};
set of ALLCOLOURS: COLOURS = ALLCOLOURS diff {BLANK};
array[COLOURS] of int: maxRun = [3,4,4,3,2,3];
```

The maximum run of each colour …

```
enum ALLCOLOURS = {grey,red,white,yellow,blue,black,BLANK};
set of ALLCOLOURS: COLOURS = ALLCOLOURS diff {BLANK};
array[COLOURS] of int: maxRun = [3,4,4,3,2,3];
```

The maximum run of each colour …

```
enum ALLCOLOURS = {grey,red,white,yellow,blue,black,BLANK};
set of ALLCOLOURS: COLOURS = ALLCOLOURS diff {BLANK};
array[COLOURS] of int: maxRun = [3,4,4,3,2,3];
```

The maximum run of each colour …

```
enum ALLCOLOURS = {grey,red,white,yellow,blue,black,BLANK};
set of ALLCOLOURS: COLOURS = ALLCOLOURS diff {BLANK};
array[COLOURS] of int: maxRun = [3,4,4,3,2,3];
```

The maximum run of each colour …

```
enum ALLCOLOURS = {grey,red,white,yellow,blue,black,BLANK};
set of ALLCOLOURS: COLOURS = ALLCOLOURS diff {BLANK};
array[COLOURS] of int: maxRun = [3,4,4,3,2,3];
```

The maximum run of each colour …

```
enum ALLCOLOURS = {grey,red,white,yellow,blue,black,BLANK};
set of ALLCOLOURS: COLOURS = ALLCOLOURS diff {BLANK};
array[COLOURS] of int: maxRun = [3,4,4,3,2,3];
```

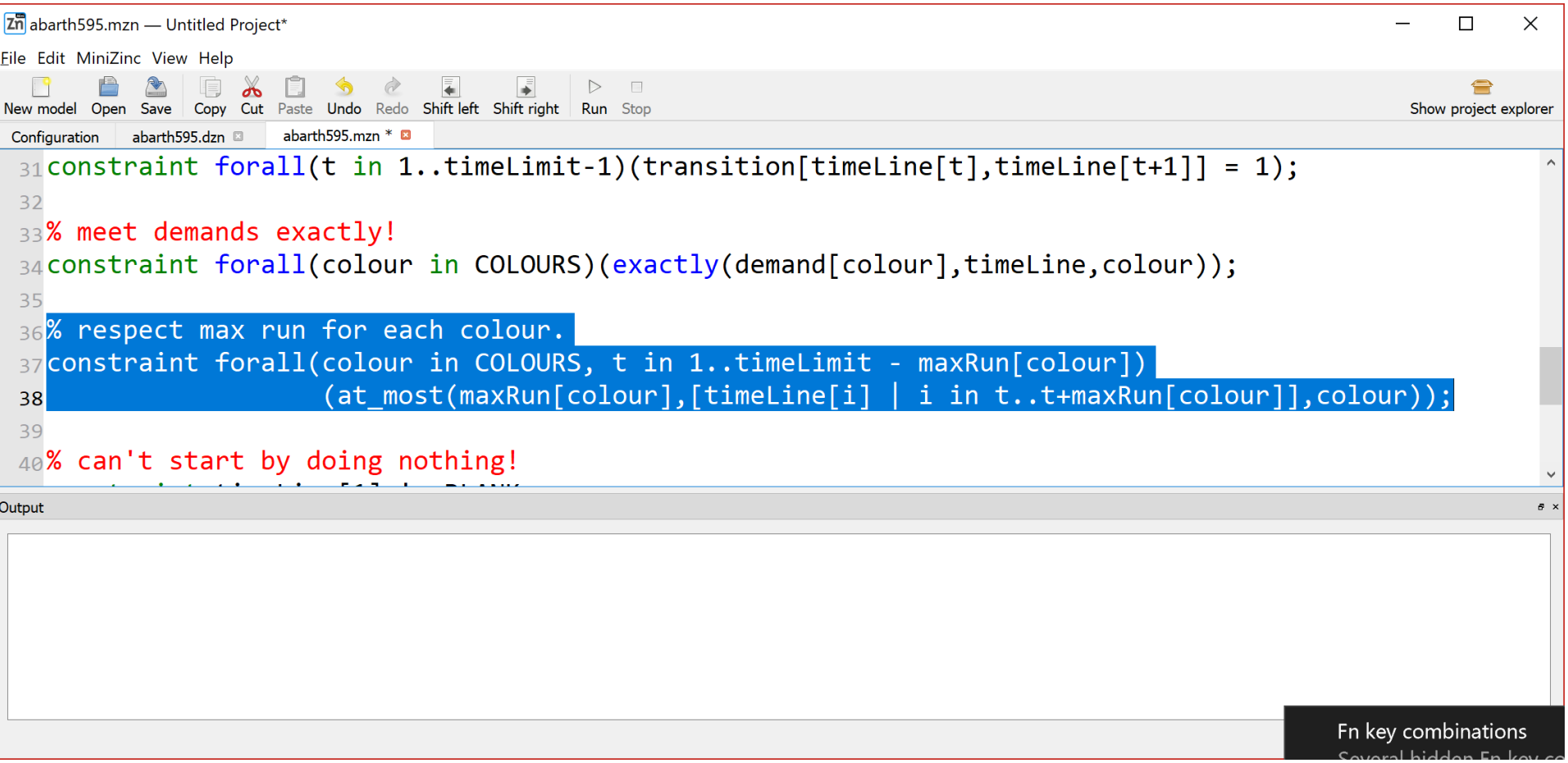
The maximum run of each colour …

```
enum ALLCOLOURS = {grey,red,white,yellow,blue,black,BLANK};
set of ALLCOLOURS: COLOURS = ALLCOLOURS diff {BLANK};
array[COLOURS] of int: maxRun = [3,4,4,3,2,3];
```

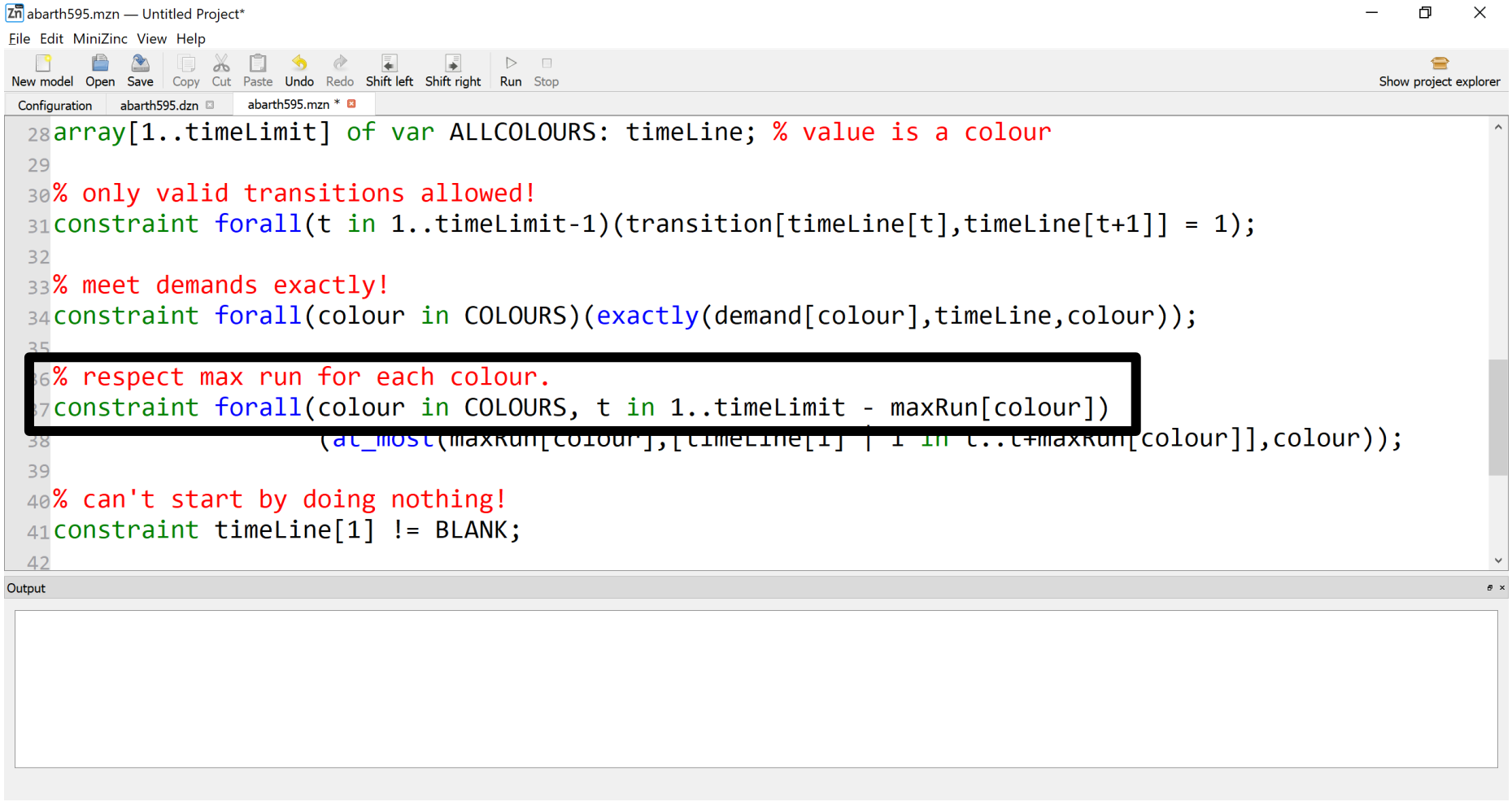
The maximum run of each colour …

```
enum ALLCOLOURS = {grey,red,white,yellow,blue,black,BLANK};
set of ALLCOLOURS: COLOURS = ALLCOLOURS diff {BLANK};
array[COLOURS] of int: maxRun = [3,4,4,3,2,3];
```

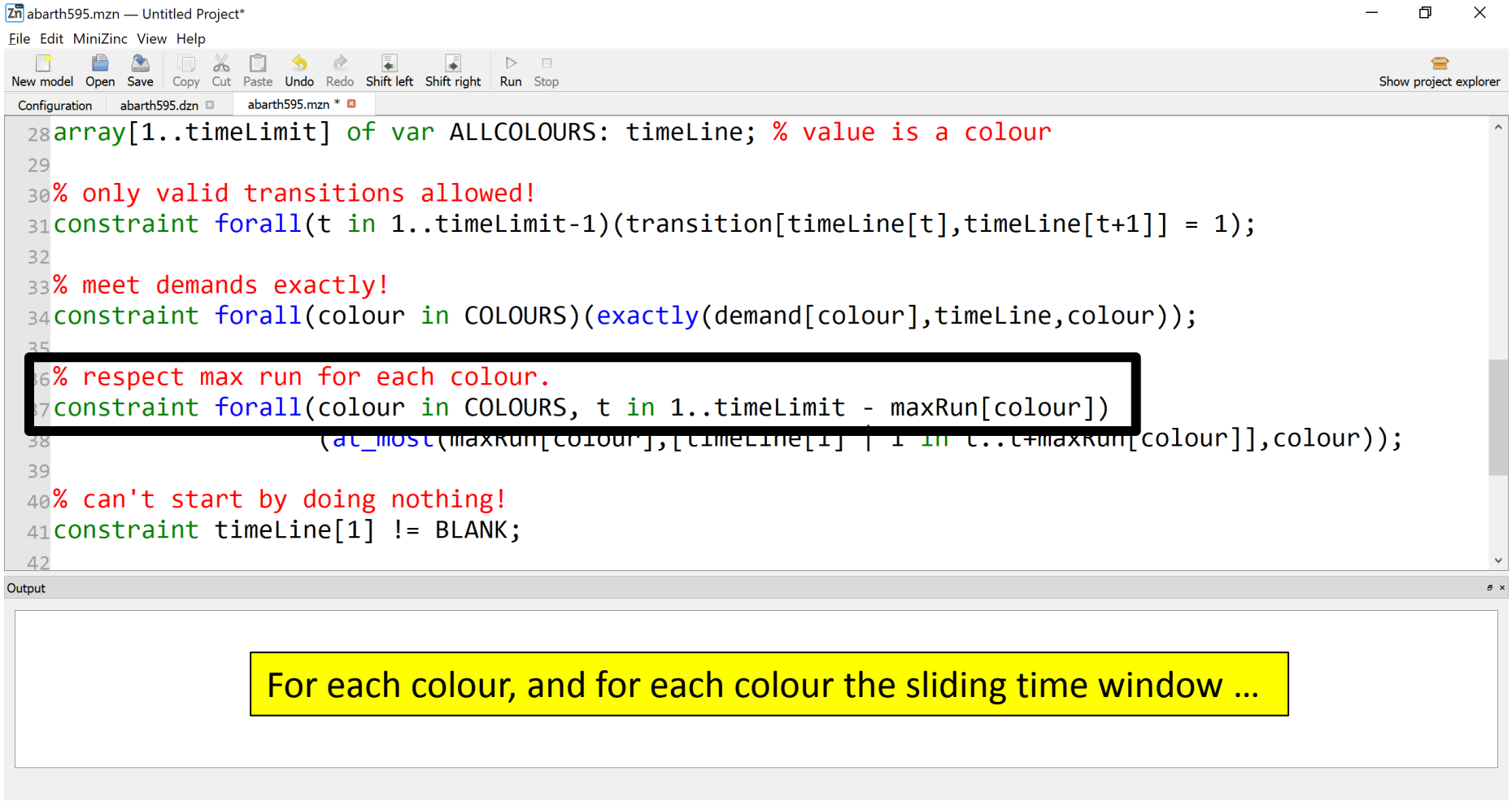
The maximum run of each colour …



The maximum run of each colour …



The maximum run of each colour …



For each colour, and for each colour the sliding time window …

The maximum run of each colour …

abarth595.mzn — Untitled Project*

File Edit MiniZinc View Help

New model Open Save Copy Cut Paste Undo Redo Shift left Shift right Run Stop Show project explorer

Configuration abarth595.dzn abarth595.mzn *

```
28 array[1..timeLimit] of var ALLCOLOURS: timeLine; % value is a colour
29
30 % only valid transitions allowed!
31 constraint forall(t in 1..timeLimit-1)(transition[timeLine[t],timeLine[t+1]] = 1);
32
33 % meet demands exactly!
34 constraint forall(colour in COLOURS)(exactly(demand[colour],timeLine,colour));
35
36 % respect max run for each colour.
37 constraint forall(colour in COLOURS, t in 1..timeLimit - maxRun[colour])
38                  (at_most(maxRun[colour],[timeLine[i] | i in t..t+maxRun[colour]],colour));
39
40 % can't start by doing nothing!
41 constraint timeLine[1] != BLANK;
42
```

Output

For that colour, and that time window starting at t …
there must be at most maxRun[colour] of that colour!

The maximum run of each colour …

abarth595.mzn — Untitled Project*

File Edit MiniZinc View Help

New model | Open | Save | Copy | Cut | Paste | Undo | Redo | Shift left | Shift right | Run | Stop | Show project explorer

Configuration | abarth595.dzn | abarth595.mzn *

```
28 array[1..timeLimit] of var ALLCOLOURS: timeLine; % value is a colour
29
30 % only valid transitions allowed!
31 constraint forall(t in 1..timeLimit-1)(transition[timeLine[t],timeLine[t+1]] = 1);
32
33 % meet demands exactly!
34 constraint forall(colour in COLOURS)(exactly(demand[colour],timeLine,colour));
35
36 % respect max run for each colour.
37 constraint forall(colour in COLOURS, t in 1..timeLimit - maxRun[colour])
38                  (at_most(maxRun[colour],[timeLine[i] | i in t..t+maxRun[colour]],colour));
39
40 % can't start by doing nothing!
41 constraint timeLine[1] != BLANK;
42
```

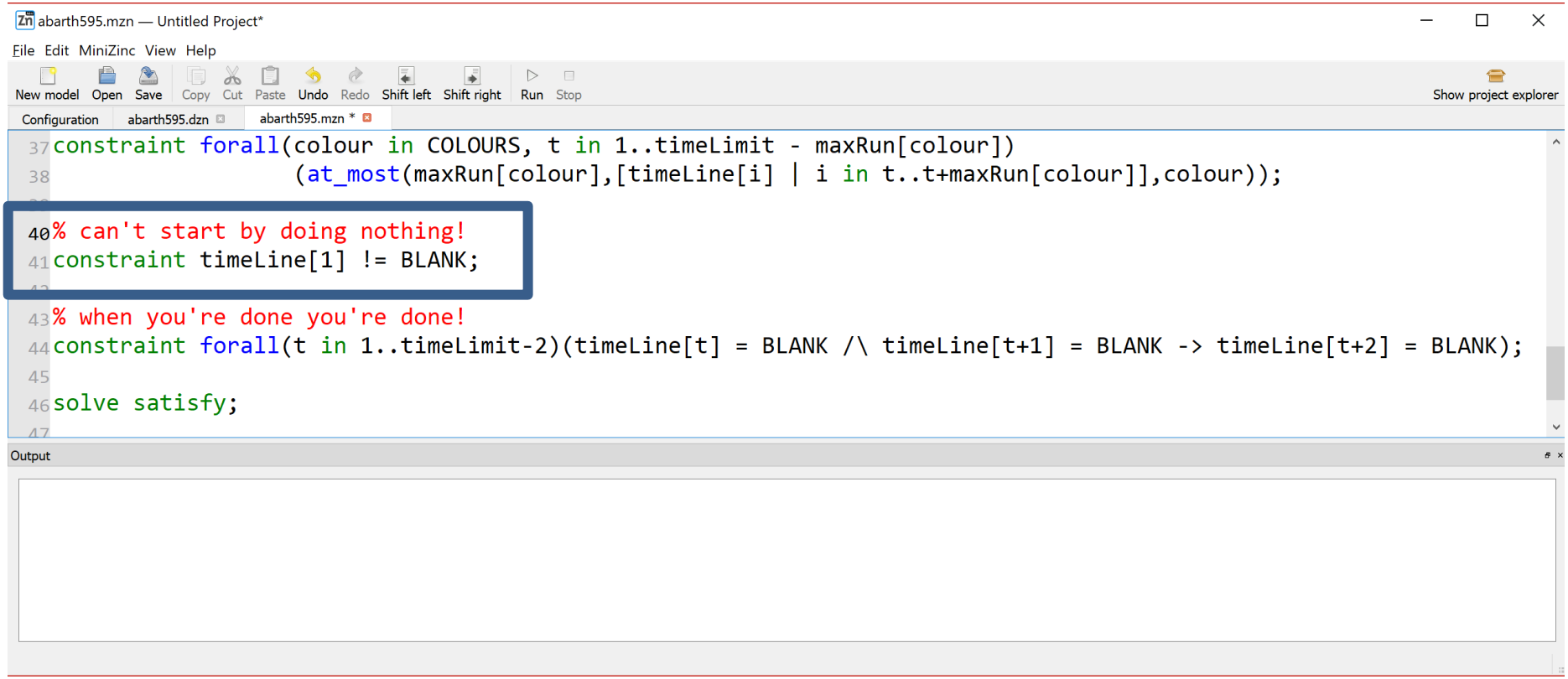
Output

NOTE: on-the-fly array comprehension ☺

Dinnae hing aboot …

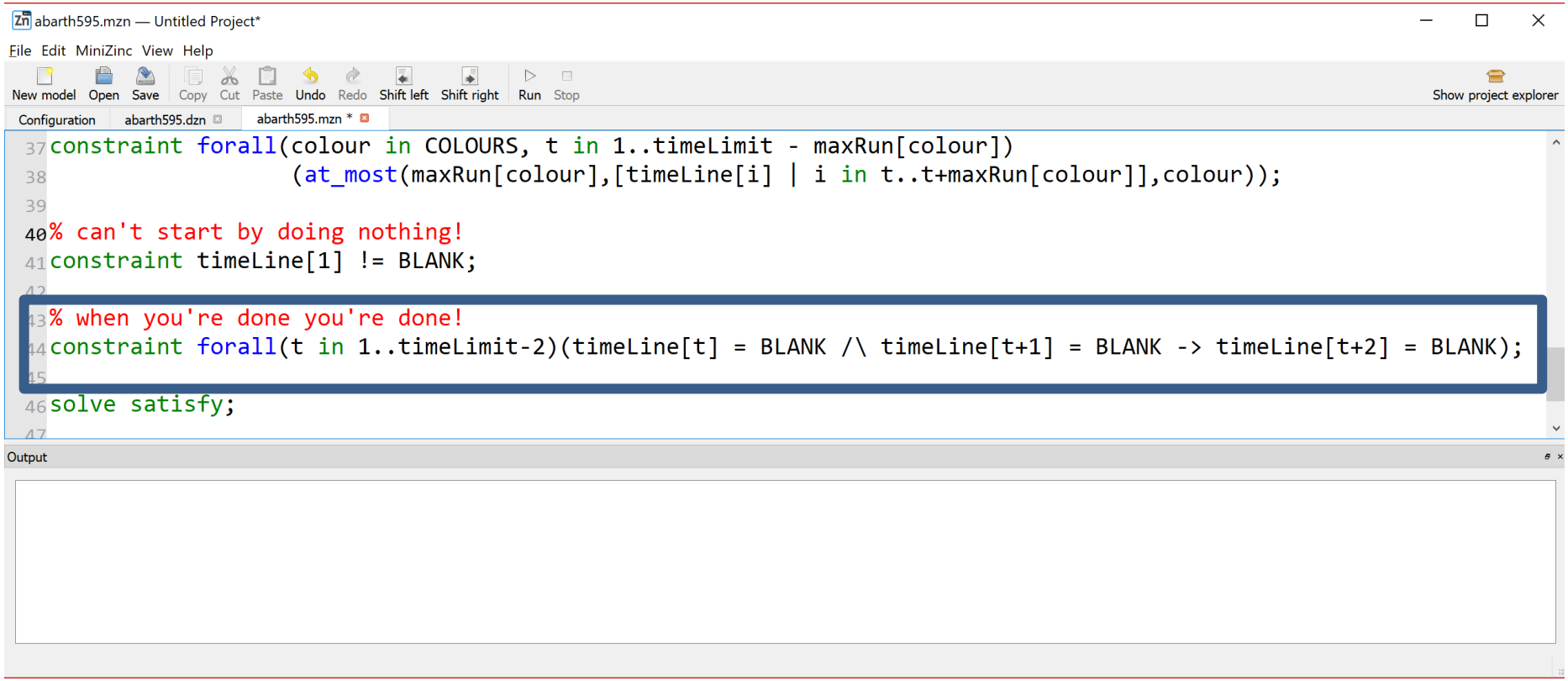
Don’t start the day by doing nothing!

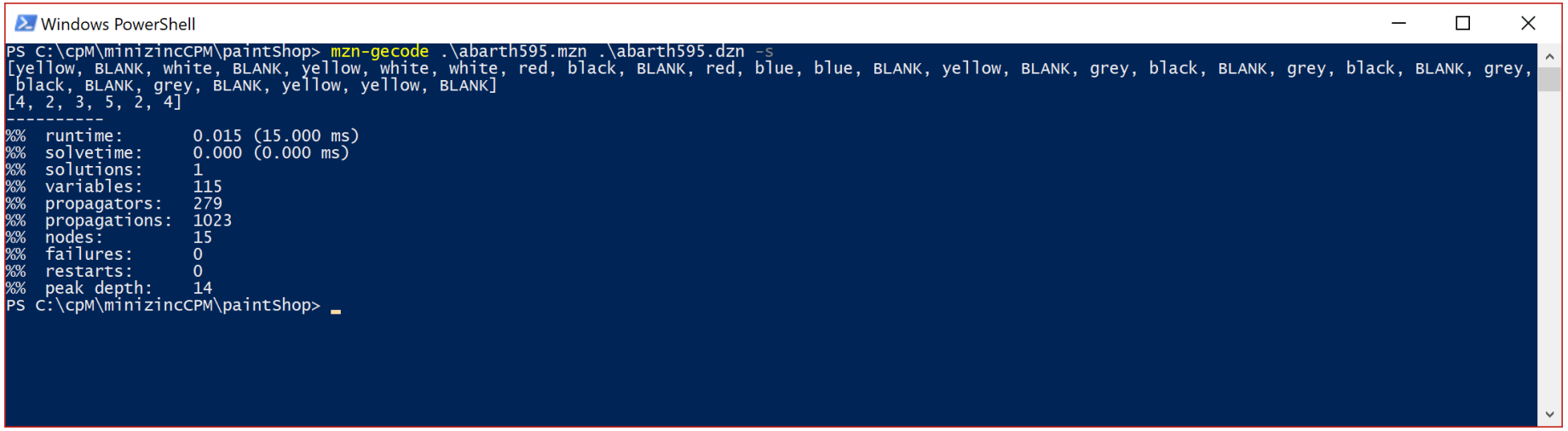
Dinnae hing aboot …



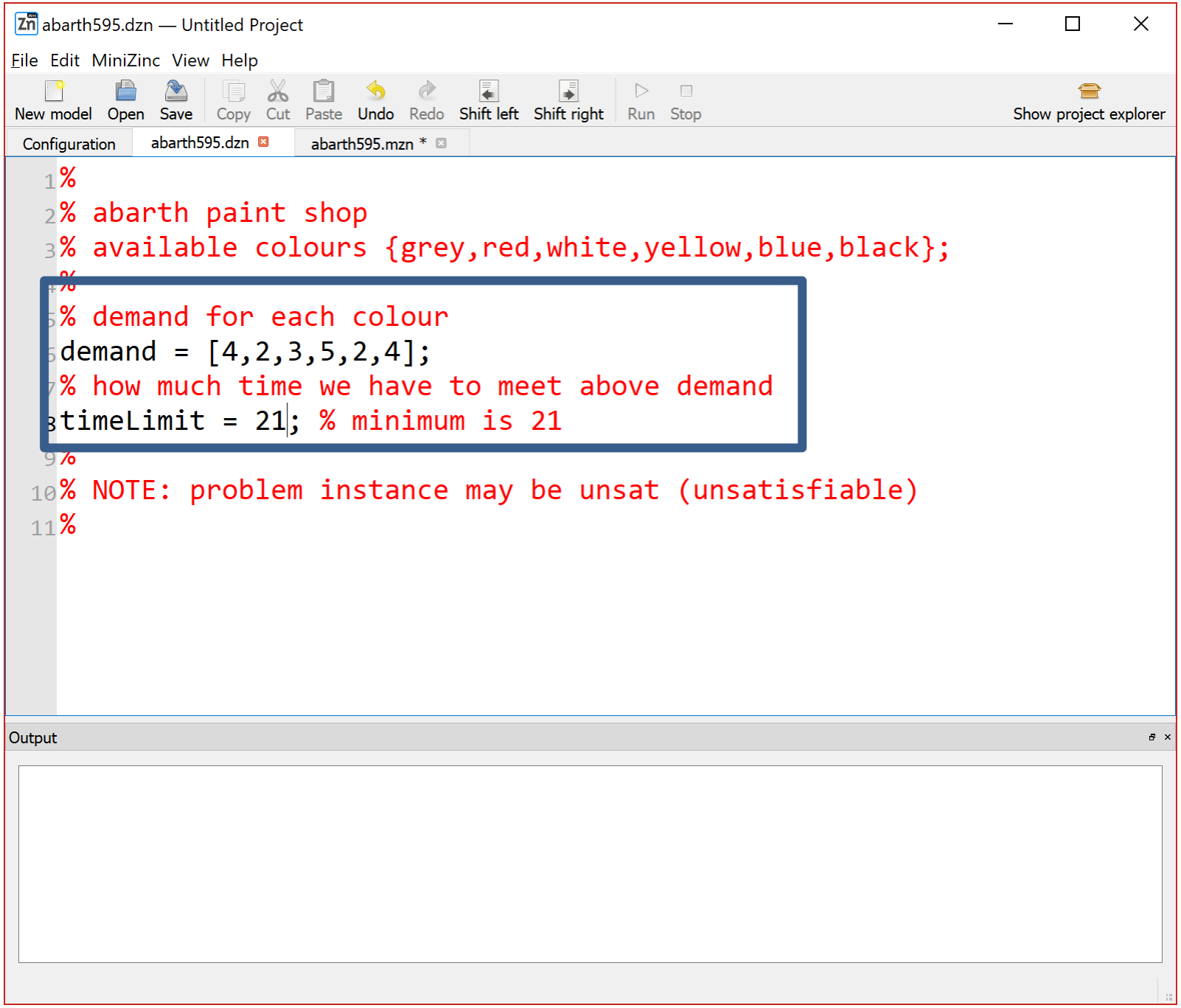
When you’re done you’re done …

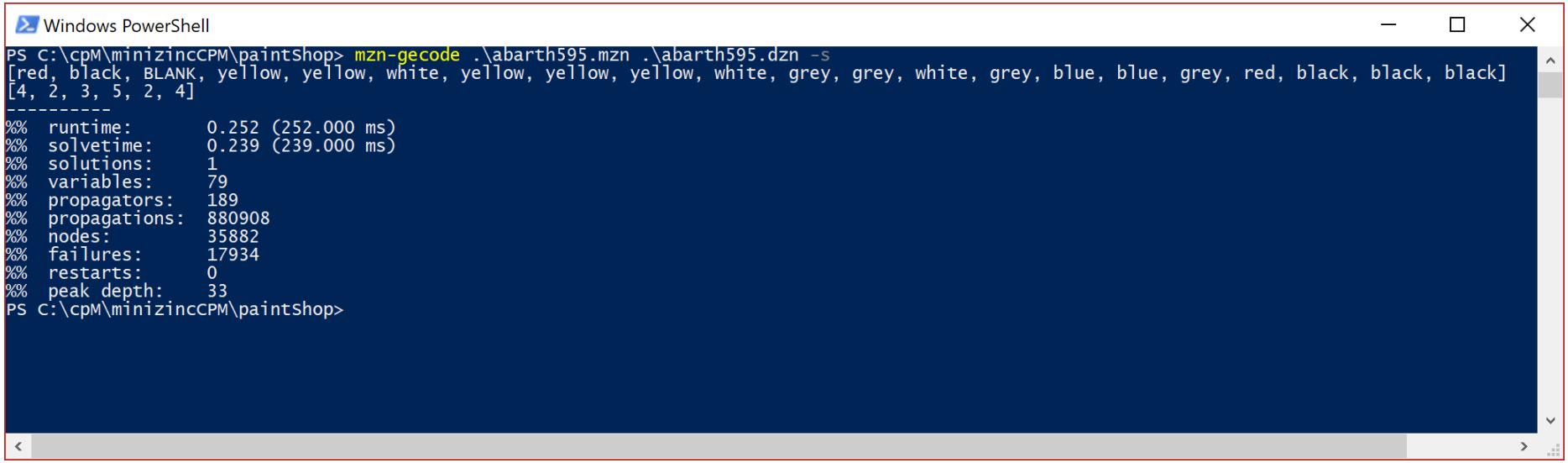
When you're done you're done …




Windows PowerShell
PS C:\cpM\minizincCPM\paintShop> mzn-gecode .\abarth595.mzn .\abarth595.dzn -s
[yellow, BLANK, white, BLANK, yellow, white, white, red, black, BLANK, red, blue, blue, BLANK, yellow, BLANK, grey, black, BLANK, grey, black, BLANK, grey, black, BLANK, grey, BLANK, yellow, yellow, BLANK]
[4, 2, 3, 5, 2, 4]

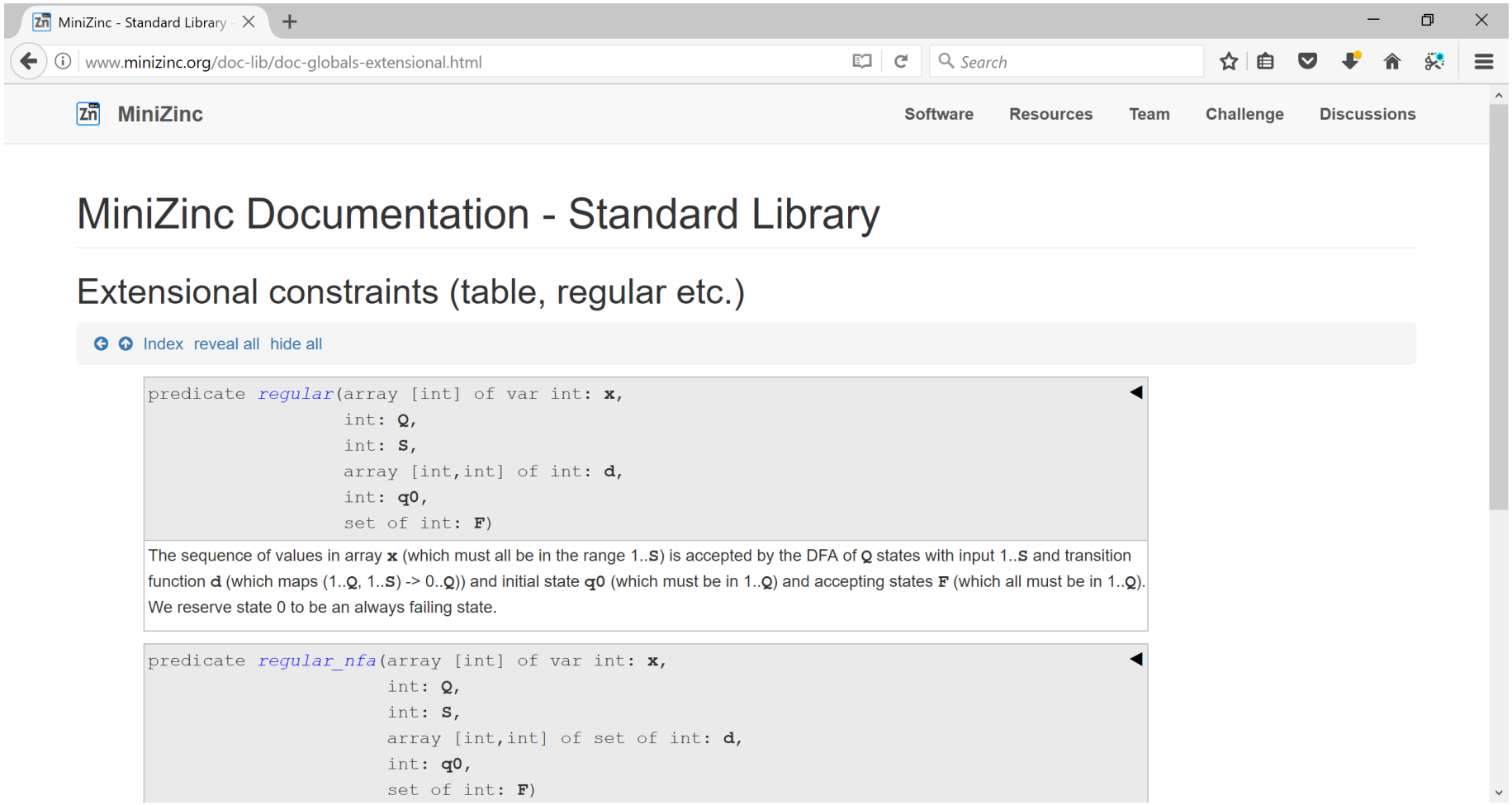
%% runtime: 0.015 (15.000 ms)
%% solvetime: 0.000 (0.000 ms)
%% solutions: 1
%% variables: 115
%% propagators: 279
%% propagations: 1023
%% nodes: 15
%% failures: 0
%% restarts: 0
%% peak depth: 14
PS C:\cpM\minizincCPM\paintShop>


abarth595.dzn — Untitled Project
File Edit MiniZinc View Help
New model Open Save Copy Cut Paste Undo Redo Shift left Shift right Run Stop Show project explorer
Configuration abarth595.dzn abarth595.mzn *
1 %
2 % abarth paint shop
3 % available colours {grey,red,white,yellow,blue,black};
4 %
5 % demand for each colour
6 demand = [4,2,3,5,2,4];
7 % how much time we have to meet above demand
8 timeLimit = 21; % minimum is 21
9 %
10 % NOTE: problem instance may be unsat (unsatisfiable)
11 %
Output

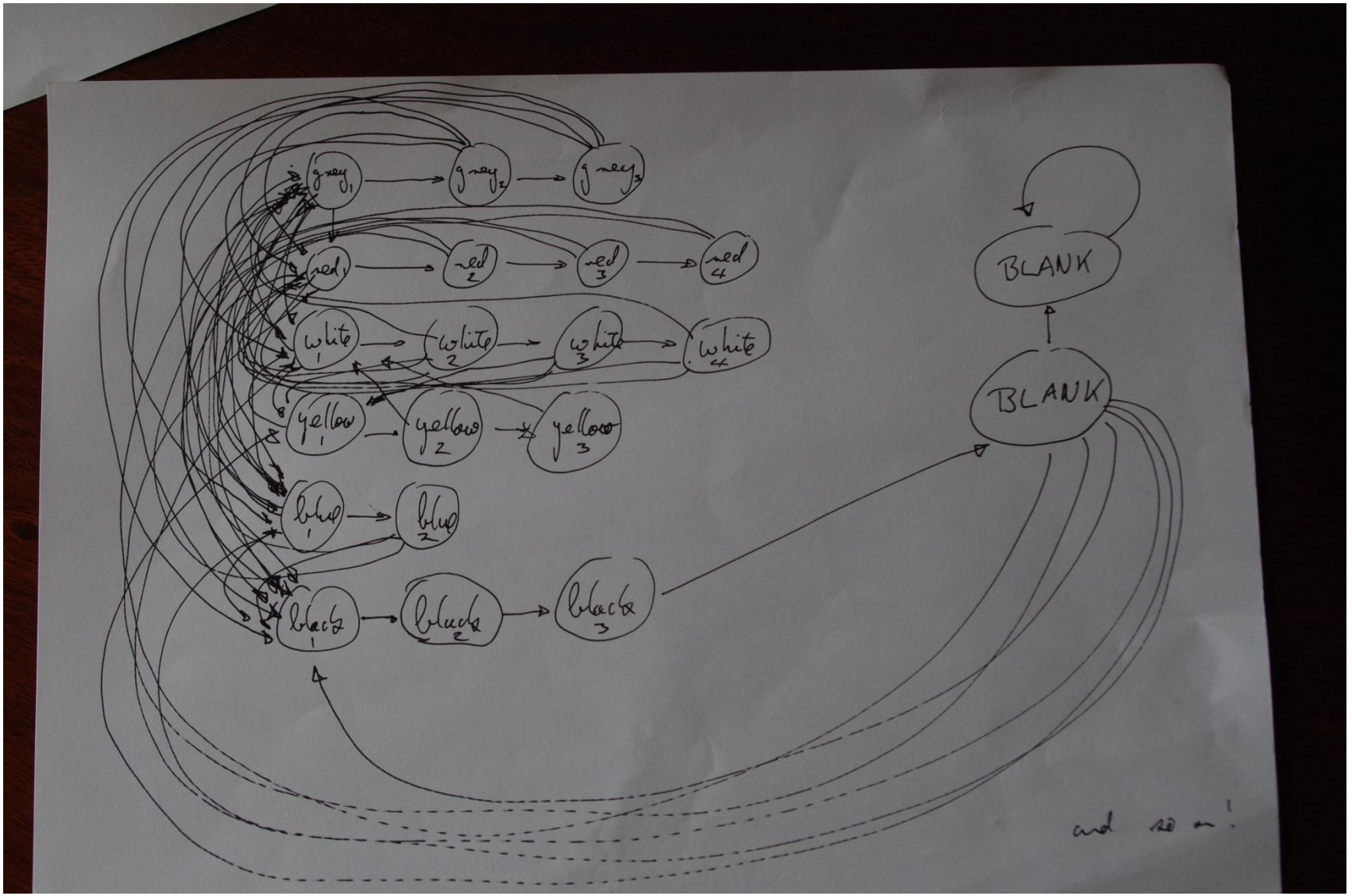

Windows PowerShell
PS C:\cpM\minizincCPM\paintShop> mzn-gecode .\abarth595.mzn .\abarth595.dzn -s
[red, black, BLANK, yellow, yellow, white, yellow, yellow, yellow, white, grey, grey, white, grey, blue, blue, grey, red, black, black, black]
[4, 2, 3, 5, 2, 4]

%% runtime: 0.252 (252.000 ms)
%% solvetime: 0.239 (239.000 ms)
%% solutions: 1
%% variables: 79
%% propagators: 189
%% propagations: 880908
%% nodes: 35882
%% failures: 17934
%% restarts: 0
%% peak depth: 33
PS C:\cpM\minizincCPM\paintShop>

Can you think of a different model?


MiniZinc - Standard Library
www.minizinc.org/doc-lib/doc-globals-extensional.html
Search
MiniZinc
Software
Resources
Team
Challenge
Discussions
MiniZinc Documentation - Standard Library
Extensional constraints (table, regular etc.)
Index reveal all hide all
predicate regular(array [int] of var int: x,
int: Q,
int: S,
array [int,int] of int: d,
int: q0,
set of int: F)
The sequence of values in array x (which must all be in the range 1..S) is accepted by the DFA of Q states with input 1..S and transition function d (which maps (1..Q, 1..S) -> 0..Q)) and initial state q0 (which must be in 1..Q) and accepting states F (which all must be in 1..Q). We reserve state 0 to be an always failing state.
predicate regular_nfa(array [int] of var int: x,
int: Q,
int: S,
array [int,int] of set of int: d,
int: q0,
set of int: F)

	gray	red	white	yellow	blue	black	BLANK
gray	1	1	1	0	1	1	1
red	0	1	0	0	1	1	1
white	1	1	1	1	0	0	1
yellow	0	0	1	1	0	0	1
blue	1	0	0	0	1	1	1
black	0	0	0	0	0	1	1
BLANK	1	1	1	1	1	1	1
Max Run	3	4	4	3	2	3	*


gray 1
gray 2
gray 3
red 1
red 2
red 3
red 4
white 1
white 2
white 3
white 4
yellow 1
yellow 2
yellow 3
blue 1
blue 2
black 1
black 2
black 3
BLANK
BLANK
and so on!

```
%
% Paint shop version Regular
%
include "globals.mzn";

enum ALLCOLOURS = {grey,red,white,yellow,blue,black,BLANK};
set of ALLCOLOURS: COLOURS = ALLCOLOURS diff {BLANK};

% input parameters
array[COLOURS] of int: demand;
int: timeLimit;

% our decision variables ... what colour (maybe BLANK) in a time slot
array[1..timeLimit] of var ALLCOLOURS: timeLine; % value is a colour

% meet demands exactly!
constraint forall(colour in COLOURS)(exactly(demand[colour],timeLine,colour));

%
% Now use a regular constraint ...
%

solve satisfy;
```