Team:
- Andrea Corradetti - andrea.corradetti2@studio.unibo.it
- Francesco Corigliano - francesco.coriglian2@studio.unibo.it

N-queens

Solutions

N	R	Rc1	Rc2	Rc3	allDiff	allDiffSym
8	92	92	92	92	92	12
9	352	352	352	352	252	46
10	724	724	724	724	724	92
12	14,200	14,200	14,200	14,200	14,200	1,787

Failures

N	R	Rc1	Rc2	Rc3	allDiff	allDiffSym
8	891	500	593	864	254	88
9	4,262	2,656	2,772	4,603	849	272
10	23,291	13,996	13,593	23,195	3722	930
12	773,550	355,041	380,595	820,127	75,823	16,224

1 What is happening when going r → rc1 → alldiff ? Why?

-R->RC1: there is no symmetry breaking here (indeed we observe the same number of solutions). RC1 is a combined model. You need to study the benefits of combined models to explain why we observe less failures in RC1 compared to R.

~~We notice less failures.~~
~~-R -> Rc1: Adding the channeling constraint, we can remove an additional wrong solution for each failure. Indeed, all wrong assignments to the rows are also invalid for the corresponding columns. We are removing some symmetric attempts.~~
-R->RC1: Combining the rows and cols models we obtain a combined model, which enhances the constraint propagation providing more scope for defining different search strategies.

-Rc1-> AllDiff: Using the global constraint, the solver can compare more than two variables at a time and detect inconsistencies earlier.

2 What is happening when going rc1 → rc2 → rc3 ? Why?

We're noticing an increase in the number of failures by removing implicit constraints.

~~-Rc1 -> Rc2 : We suppose that, without allDifferent, we can't immediately prune row i and column j from the domain after placing a queen in cell i,j. After placing the first queen on column i (X1=i) , the solver might place a second queen on the same row but on a different column (X2 = i). At this point the solver realizes that the channeling constraint can't be satisfied for Y.~~

-RC1->RC2: no, this cannot happen due to the channeling constraints (X1 = i enforces that Yi = 1, so we cannot have Yi =2 anymore and therefore we cannot have X2 = i either). Indeed, that's why we are able to drop all the alldifferent constraints.

By using the channeling constraint, the model is still capable of respecting the limit of one queen for each row/col. Inverse functions can't have more than one Y for an X and vice versa.
But the global constraint is capable of further shrinking the search size using ad hoc implementations, that's why in RC2 we notice more failures.
The same condition would be maintained in the end, but the global constraints affords stronger propagation.

-Rc-2 -> Rc-3
~~We have removed the diagonal attack constraint on the columns array.~~
~~If we remove the constraint on the diagonals for the columns array, we get no pruning when assigning col variables. Only after assigning the corresponding row variable according to the channeling constraint, we get the desired domain restriction. If the solver decides to assign Y variables before X variables, the search space is larger for the next assignment.~~

-RC2 -> RC3: again, thanks to the channeling constraints, with every assignment we will get the desired effect.
Removing the diagonal constraint on the columns array, we are noticing an important increase in the number of failures. That's because we are removing an implied constraint.

-What you got right is that the removed constraints are indeed implicit constraints. As you can see, their removal is worsening the models, which means they were useful. How do we call in our lectures the implicit constraints that turn out to be useful? (note that we use a different terminology for the implicit constraints that are useless).

-Implied constraints are useful: even though they are semantically redundant, they are computationally impactful (they reduce the search space size). Redundant constraints, on the other hand, are useless.

3 What is happening when going alldiff → alldiffsym? Why?

We linked the boolean matrix to the one dimensional array to express constraints on symmetric solutions. Doing so we can keep only the smallest solutions in terms of lexicographic order. This allows us to remove 7 further symmetric assignments for each failure.
3. alldiff -> alldiffsym: we not only remove symmetric failures, but also symmetric consistent partial assignments and symmetric solutions. That is, we remove all symmetric variants of all possible search states. That's why we also reduce the size of the search space.

Sequence puzzle

N	Base		Base + Implied	
	Fails	Time	Fails	Time
500	617	11,764 s	495	7,543 s
1000	1247	1m 25s	995	36,199 s

– What is happening when going base → base+implied ? Why?
~~The base constraint can only verify the entirety of the array once all variables are assigned. The implied constraints can catch mistaken assignments already on the second assignment.~~

No, what you are saying is not correct. In CP, none of the constraints are checked once all variables are assigned (you can fail this course with this statement 😃) We have the concept of constraint propagation which prunes inconsistent values from the domains of the unassigned variables at each variable assignment.

I think you didn't get right the concept of implied constraints either. Please study well the modelling topics and revisit the questions.

The solver cannot deduce from the main constraint that those 2 implied constraints must hold. The properties will hold in the final solutions but the solver can't use them to prune the search tree (semantically redundant but computationally relevant). By making them explicit we can further reduce the search space.