Institut de Recherche en Informatique de Toulouse
CNRS - INP - UT3 - UT1 - UT2J

Toulouse, January 2, 2020

TouIST tutorial #3: Introduction to generalized connectors

1 Prerequisites

You must have done the previous tutorials. You must know how to model the problem of the Latin Square and interpret its solutions.

2 Goal of the tutorial

At the end of this tutorial, you will be able to use the generalized connectors in order to make your formulas much more compact. You will also learn how to define simple variables or lists in order to use them in conjunction with generalized operators. You will be able to make the length of your model (almost) independent of the size of the modeled problem.

3 Back to the 3×3 Latin square

Recall a Latin square 3×3 is a grid of 3×3 cells. Each of the 9 cells must contain a value in $VAL = \{1, 2, 3\}$, and only one. The boxes on the same line all contain different values. The boxes in the same column all contain different values.

The constraints are the following:

1. each cell contains *at least* one value in VAL;
2. each cell contains *at most* one value in VAL;
3. each value of VAL appears at most once on the same *line* (note that, together with constraint 1, each value appears at least once on the same line);
4. each value of VAL appears at most once on the same *column* (note that, together with constraint 1, each value appears at least once on the same column);

Our running example is always the following initial state:

2		1
3		

Again, for each of the cells, you can define three propositional variables corresponding to the three possible values $\{1, 2, 3\}$. Thus, $C(i, j, k)$ will represent the proposition "The coordinate cell (i, j) contains the value k", and $\neg C(i, j, k)$ will represent the proposition "The coordinate cell (i, j) does not contain the value k".

Exercise 1 *Give the modeling of the initial state.*

Exercise 2 *Using* only one formula *of type $C(i, j, n)$, encode the first constraint. (Do not be afraid, below, we guide you step by step to achieve this result!)*

How to simplify the formula for a single cell? Recall that, to impose that the cell $(2, 1)$ contains at least one value in VAL, we wrote:

```
C(2,1,1) or C(2,1,2) or C(2,1,3)
```

Note that the number in red (the value in the $(2, 1)$ cell) is the only parameter that changes from one atomic variable to another. So TouIST lets you write this formula like this:

```
bigor $v in [1,2,3]:
     C(2,1,$v)
end
```

`$v` is a TouIST variable and `[1,2,3]` is a list of indexes. We can also define a variable for this list:

```
$VAL = [1,2,3]
bigor $v in $VAL:
     C(2,1,$v)
end
```

Mathematically, the previous formula is written:

$$VAL = \{1, 2, 3\}$$
$$\bigvee_{v \in VAL} C(2, 1, v)$$

We use here only one occurrence of the variable $C(i, j, v)$ ... but we only describe the values of the cell $(2, 1)$.

How to simplify the formula for a same line? For describing the possible values of the other cells of the line 2, we would have to write something like:

$$VAL = \{1, 2, 3\}$$
$$\left(\bigvee_{v \in VAL} C(2, 1, v)\right) \wedge \left(\bigvee_{v \in VAL} C(2, 2, v)\right) \wedge \left(\bigvee_{v \in VAL} C(2, 3, v)\right)$$

where the column number (in blue) is the only number that changes. So, if we define j as a new variable for column indexes, we can write:

$$IND = \{1, 2, 3\}$$
$$VAL = \{1, 2, 3\}$$
$$\bigwedge_{j \in IND} \bigvee_{v \in VAL} C(2, j, v)$$

which can be translated in TouIST as follows:

```
$IND = [1,2,3]
$VAL = [1,2,3]
bigand $j in $IND:
   bigor $v in $VAL:
      C(2,$j,$v)
   end
end
```

Question 2.1 *It's up to you to generalize the formula above to describe the fact that* all cells in the grid *have at least a value of VAL.*

Remark 1 *It is important to note two things: 1) the formula you have just written (1 propositional variable and 3 generalized operators) allows you to summarize the 9 formulas (27 propositional variables and 18 logical operators) that you had needed in the previous tutorial:*

```
C(1,1,1) or C(1,1,2) or C(1,1,3)
C(1,2,1) or C(1,2,2) or C(1,2,3)
C(1,3,1) or C(1,3,2) or C(1,3,3)
C(2,1,1) or C(2,1,2) or C(2,1,3)
C(2,2,1) or C(2,2,2) or C(2,2,3)
C(2,3,1) or C(2,3,2) or C(2,3,3)
C(3,1,1) or C(3,1,2) or C(3,1,3)
C(3,2,1) or C(3,2,2) or C(3,2,3)
C(3,3,1) or C(3,3,2) or C(3,3,3)
```

2) to switch to a 4×4 grid, simply add the number 4 to the variables IND and VAL, without changing anything else! (While in the code above, we would have gone from 9 to 16 formulas, that is, 48 propositional variables and 24 logical operators!)

Exercise 3 *Now write the formula to force each cell to have at most one value (second constraint).*

Recall that, if a cell has a certain value, then it does not have the other two possible values. Here too you will have help (but a little less!).

Recall that such a constraint only for cell $(1, 1)$ was written (in the previous tutorial):

Université Paul Sabatier, IRIT / ADRIA-LILaC
118 route de Narbonne, 31062 Toulouse cedex 9

```
C(1,1,1) => not C(1,1,2) and not C(1,1,3)
```

that can be written in propositional logic:

$$C(1,1,1) \rightarrow \neg C(1,1,2) \wedge \neg C(1,1,3)$$

that could be rewritten (considering that the numbers in red are all the numbers of $VAL \backslash \{1\}$):

$$C(1,1,1) \rightarrow \bigwedge_{\substack{v_2 \in VAL \\ v_2 \neq 1}} \neg C(1,1,v_2)$$

Good new: you can write such an expression with TouIST!

```
C(1,1,1) => bigand $v2 in $VAL when $v2!=1:
                  not C(1,1,v2)
            end
```

Question 3.1 *Generalize the previous expression to express the fact that, for any value v_1 of cell $(1,1)$, this cell has no value v_2 different from v_1.*

Question 3.2 *And now generalize this last expression to express the fact that, for any value v_1 of cell (i,j) where $i,j \in VAL$, this cell has no value v_2 different from v_1.*

Congratulations! You have just written 1 formula (2 instances of propositional variables, 4 generalized operators, 2 logical operators) which replaces no less than 27 formulas (81 instances of propositional variables, 108 operators) from the previous model!

Exercise 4 *Write the constraint (set of formula) saying that each value between 1 and 3 only appears once and only once on each line.*

(Note: this means that if a cell has a certain value, the other two cells on the same row cannot have this value.)

Exercise 5 *Finally, write the constraint (set of formula) indicating that each value between 1 and 3 only appears once and only once on each column.*

Exercise 6 *Press* Solve*: you should obtain 1 model. Write the corresponding square.*

Exercise 7 *In the modeling of the previous tutorial, we had:* $9 + 3 * 27 = 90$ *formulas,* $27 + 3 * 81 = 270$ *instances of propositional variables,* $18 + 3 * 108 = 342$ *logical operators. What are these numbers for your current modeling?*

4 For further

In this section, we suggest some improvements in order to: generalize the code for a Latin square of any dimension N; make the writing of generalized operators nested one inside the other even more compact; optimize the code by avoiding repeatedly producing the same constraints.

4.1 From a square 3×3 to a square $N \times N$

Currently, the size of the grid is determined by the set of indices $IND = \{1, 2, 3\}$ and values $VAL = \{1, 2, 3\}$. TouIST also allows you to define ranges of values such as [1..3], which allows you to write:

```
;; square dimension
$N = 3

;; a set of cells indexes
$IND = [1 .. $N]

;; a set of cells values
$VAL = [1 .. $N]
```

Just change the value of N, and your model can solve a Latin square of any (finite) dimension without changing anything else!

Exercise 8 *Generalize your* TouIST *program to any square of dimension N and try $N = 5$. Check that the first solution obtained is correct.*

4.2 Nesting of generalized operators

Mathematically, for every $n \in \mathbb{N}$:

$$\bigwedge_{i_1 \in S_1} \bigwedge_{i_2 \in S_2} \cdots \bigwedge_{i_n \in S_n} P(i_1, i_2, ..., i_n)$$

is equivalent to:

$$\bigwedge_{\substack{i_1 \in S_1 \\ i_2 \in S_2 \\ \cdots \\ i_n \in S_n}} P(i_1, i_2, ..., i_n)$$

(The conjuction of $P(i_1, i_2, ..., i_n)$ for $(i_1, i_2, ..., i_n) \in S_1 \times S_2 \times \cdots \times S_n$.)

Good news! You can write that in TouIST. For instance:

```
bigand $i in $IND:
      bigand $j in $IND:
            bigor $v in $VAL:
                  C($i,$j,$v)
            end
      end
end
```

can be rewritten:

```
bigand $i, $j in $IND, $IND:
     bigor $v in $VAL:
          C($i,$j,$v)
     end
end
```

Exercise 9 *Rewrite the constraints using the compact writing above. Make sure you get the same models as before.*

4.3 Optimizing the code

In the previous tutorial, the third constraint (each line constains at least 1 occurrence of each value in *VAL*) begun as follow:

```
C(1,1,1) => not C(1,2,1) and not C(1,3,1)
C(1,1,2) => not C(1,2,2) and not C(1,3,2)
C(1,1,3) => not C(1,2,3) and not C(1,3,3)
C(1,2,1) => not C(1,1,1) and not C(1,3,1)
C(1,2,2) => not C(1,1,2) and not C(1,3,2)
C(1,2,3) => not C(1,1,3) and not C(1,3,3)
C(1,3,1) => not C(1,1,1) and not C(1,2,1)
C(1,3,2) => not C(1,1,2) and not C(1,2,2)
C(1,3,3) => not C(1,1,3) and not C(1,2,3)
```

Note that for the (red) first formula, we have the following equivalences:

$$\begin{aligned} & C(1,1,1) \rightarrow \neg C(1,2,1) \wedge \neg C(1,3,1) \\ \equiv & \big(C(1,1,1) \rightarrow \neg C(1,2,1)\big) \wedge \big(C(1,1,1) \rightarrow \neg C(1,3,1)\big) \\ \equiv & \big(C(1,2,1) \rightarrow \neg C(1,1,1)\big) \wedge \big(C(1,3,1) \rightarrow \neg C(1,1,1)\big) \end{aligned}$$

In others words, the above set of formulas can simplified as follows:

```
C(1,1,1) => not C(1,2,1) and not C(1,3,1)
C(1,1,2) => not C(1,2,2) and not C(1,3,2)
C(1,1,3) => not C(1,2,3) and not C(1,3,3)
C(1,2,1) => not C(1,3,1)
C(1,2,2) => not C(1,1,2) and not C(1,3,2)
C(1,2,3) => not C(1,1,3) and not C(1,3,3)
C(1,3,1) => not C(1,2,1)
C(1,3,2) => not C(1,1,2) and not C(1,2,2)
C(1,3,3) => not C(1,1,3) and not C(1,2,3)
```

Exercise 10 *Similarly, simplify the set of formulas above as much as possible. What do you see between the column indices to the left and to the right of the symbol* => *? What can you deduce about the modification to be made to your 3rd constraint to produce a minimal (optimized) set of formulas? Test that the resulting* TouIST *program is still correct.*

Exercise 11 *Do a similar optimization for the 2nd constraint and test your new program* TouIST.

Exercise 12 *Do a similar optimization for the 4th constraint and test your new program* TouIST.

Exercise 13 *Generate a square with the same initial state* $\{C(1,1,2), C(1,3,1), C(3,1,3)\}$. *The first solution should be:*

2	9	1	8	7	6	5	4	3
9	8	7	6	5	4	3	2	1
3	7	9	5	8	2	6	1	4
8	6	5	9	4	7	1	3	2
7	5	8	4	1	3	2	6	9
6	4	3	7	2	1	8	9	5
4	3	2	1	6	5	9	7	8
5	1	6	2	3	9	4	8	7
1	2	4	3	9	8	7	5	6

Authors: **Olivier GASQUET** UPS-IRIT/LILaC — **Dominique LONGIN** CNRS-IRIT/LILaC — **Frédéric MARIS** UPS-IRIT/ADRIA