Global Inverse Consistency for Interactive Constraint Satisfaction

Christian Bessiere[1] Hélène Fargier[2] Christophe Lecoutre[3]

[1]LIRMM-CNRS, University of Montpellier, France

[2]IRIT-CNRS, University of Toulouse, France

[3]CRIL-CNRS, University of Artois, Lens, France

International Conference CP – September 20th, 2013

Project ANR "BR4CP" – Project OSEO "Pajero"

Interactive Constraint Satisfaction

Some domains require interactive solving. For example:

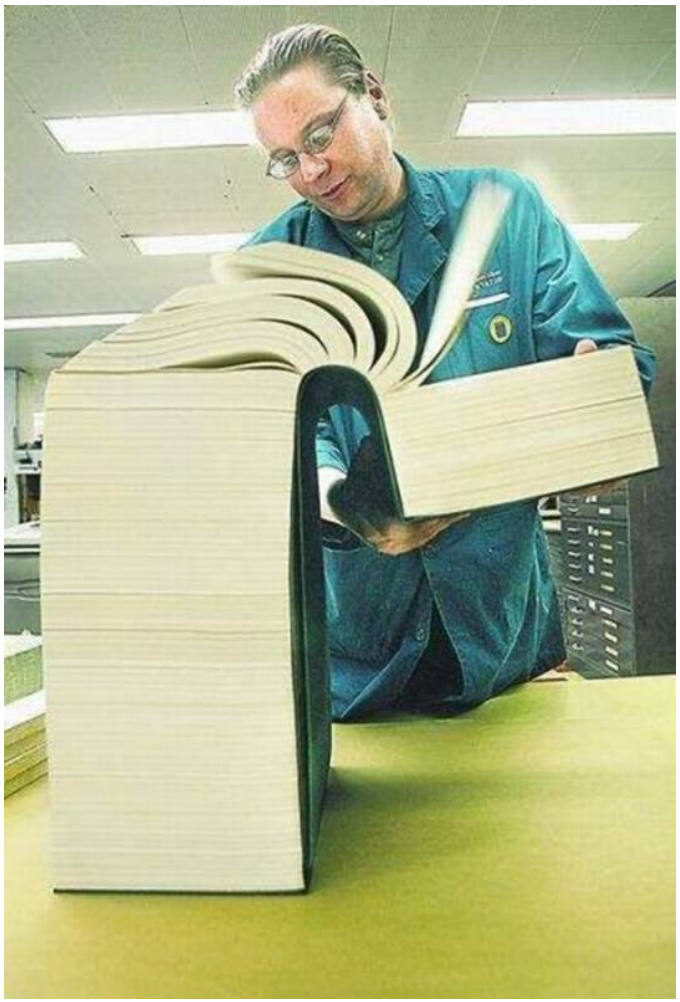
- protein design (Synthia system)
- product configuration (constraint-based models)

Among configuration applications:

- computers
- cars
- kitchen
- travel packages
- ...

Important: interactions/preferences ($\approx$ branching decisions) are made/expressed by the user

Configuration

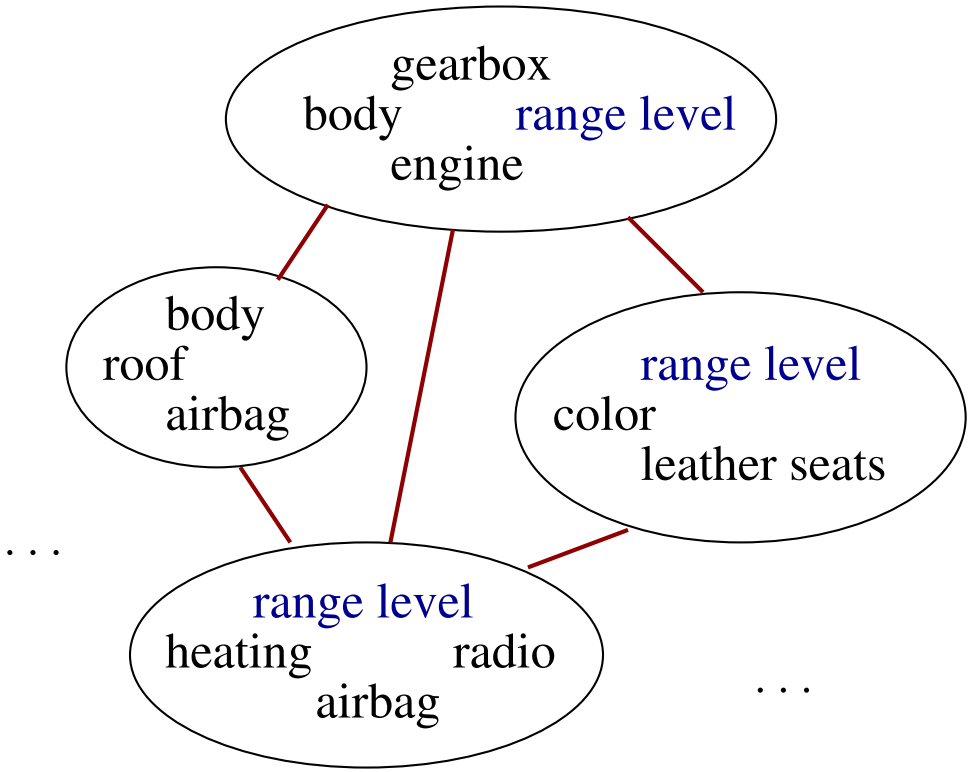


In the domain of configuration, a catalog of possible products may contain billions of solutions.

So the catalog must be represented as:

- a MDD (but limited type of interactions)
- or a constraint network (any type of interactions)

Illustration: catalog of car models as a CN



gearbox	engine	body	range
5	V8	coupe	classic
6	V10	cabriolet	luxary

color	leather seats	range
red	yes	classic
yellow	no	classic

Issue of Interactive solving

Interactive solving

Preferences of the user: any new constraints (e.g., instantiations of variables)

But what about conflicting preferences/decisions, i.e., possible dead-ends?

- *either explain them, or*
- *avoid them*

Explaining dead-ends

Everytime a dead-end occurs, one may compute a minimal explanation (minimal unsatisfiable core).

But the user may be unhappy with that.

> *Why this system forces me to retract one of my choices/preferences?*

Avoiding dead-ends

Enforcing/maintaining some level of consistency?

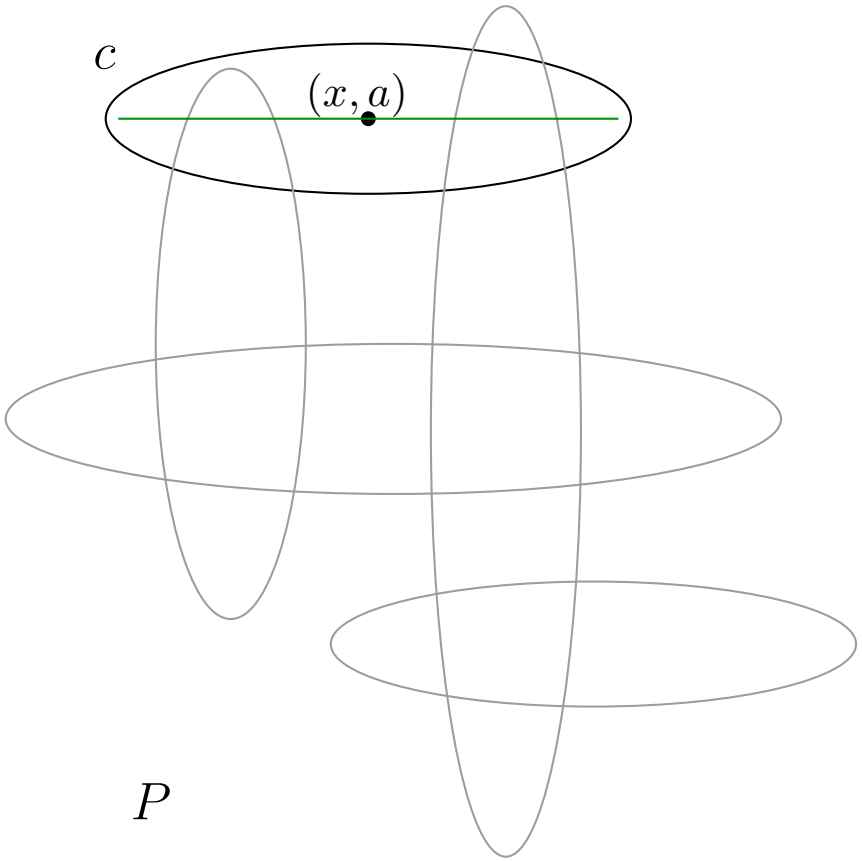
- GAC (the typical choice)
- ESAC
- SAC
- weak k-SAC
- maxRPWC

However,

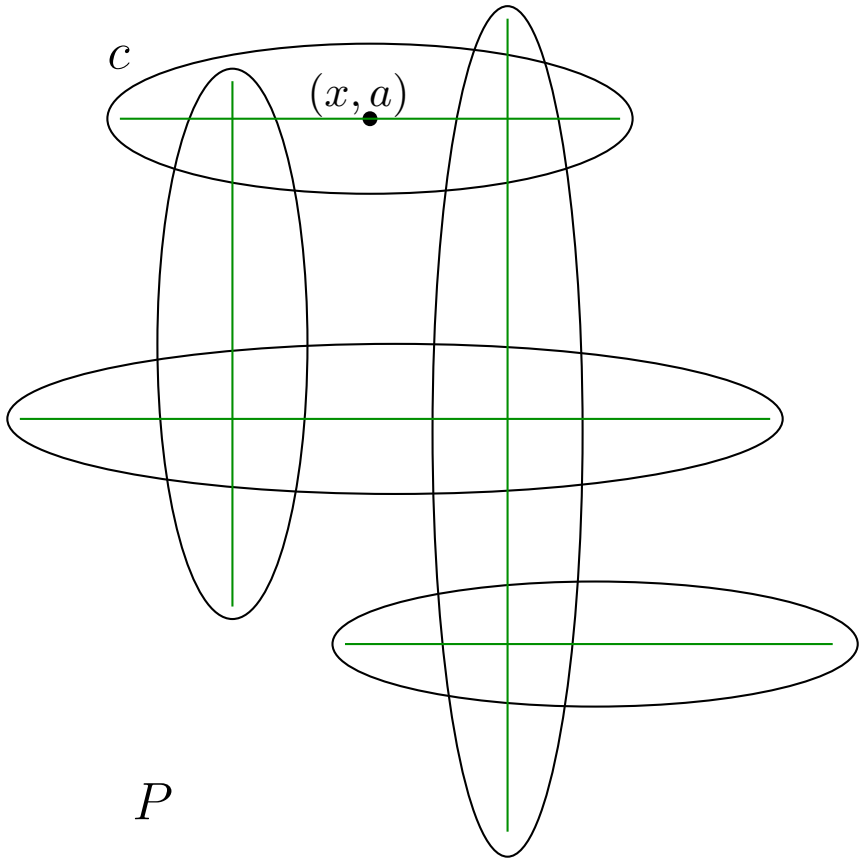
- no guarantee is given
- the experimental study is not easy

Fortunately: we show that it is possible to avoid failures on many problems by maintaining a very strong property called *global inverse consistency* (GIC).

GAC versus GIC



A value (x, a) is GAC for a constraint c iff there exists a "solution" of c involving (x, a).



A value (x, a) is GIC for a CN P iff there exists a solution of P involving (x, a).

GIC

Definition (Global Inverse Consistency)

- A value (x, a) of a CN P is *globally inverse consistent (GIC)* iff $\exists I \in sols(P) \mid I[x] = a$.
- A CN P is GIC iff every value of P is GIC.

The *GIC closure* of a CN P is the CN obtained from P by removing all the values that are not GIC.

Complexity Results

Theorem

Deciding whether a constraint network P is GIC is NP-complete, even if P is satisfiable.

Theorem

Computing the GIC closure of a constraint network P is NP-hard and NP-easy, even if P is satisfiable.

Theorem

Generating a solution to a GIC constraint network cannot be done in polynomial time, unless $P = NP$.

Outline

GIC1 - our baseline

Algorithm 1: GIC1(P: CN)

foreach *variable* $x \in vars(P)$ **do**
    foreach *value* $a \in dom(x)$ **do**
        $I \leftarrow$ searchSolutionFor$(P|_{x=a})$
        if $I = nil$ **then**
            remove a from $dom(x)$
            if $dom(x) = \emptyset$ **then**
                return *false*

return *true*

GIC2, GIC3 and GIC4

The second algorithm, GIC2, uses timestamping. It allows us to use the multidirectionality of solutions.

The third algorithm, GIC3, additionnally uses residues, which correspond to solutions that have been previously found.

A flavour of AC3^{rm}

The fourth algorithm, GIC4, records current solutions in a table.

A flavour of STR

GIC4 - part 1/3

Algorithm 2: GIC4(P: CN)

```
// Initialization of structures
```
$S^{val} \leftarrow \emptyset$

foreach *variable* $x \in vars(P)$ **do**
  if $|dom(x)| \neq$ `lastSize`$[x]$ **then**
    $S^{val} \leftarrow S^{val} \cup \{x\}$

$S^{sup} \leftarrow \emptyset$

foreach *variable* $x \in vars^{fut}(P)$ **do**
  `gicValues`$[x] \leftarrow \emptyset$
  $S^{sup} \leftarrow S^{sup} \cup \{x\}$

...

Algorithm 3: GIC4(P: CN)

```
...
// The table of current solutions is traversed
i ← 1
while i ≤ nbSolutions do
    if isValid(solutions[i]) then
        handleSolution(solutions[i])
        i++
    else
        solutions[i] ← solutions[nbSolutions]
        nbSolutions − −
...
```

GIC4 - auxiliary functions

Algorithm 4: handleSolution(I : solution)

foreach *variable* $x \in S^{sup}$ **do**
    if $I[x] \notin \texttt{gicValues}[x]$ **then**
        $\texttt{gicValues}[x] \leftarrow \texttt{gicValues}[x] \cup \{I[x]\}$
        if $|\texttt{gicValues}[x]| = |dom(x)|$ **then**
            $S^{sup} \leftarrow S^{sup} \setminus \{x\}$

Algorithm 5: isValid(I : instantiation): Boolean

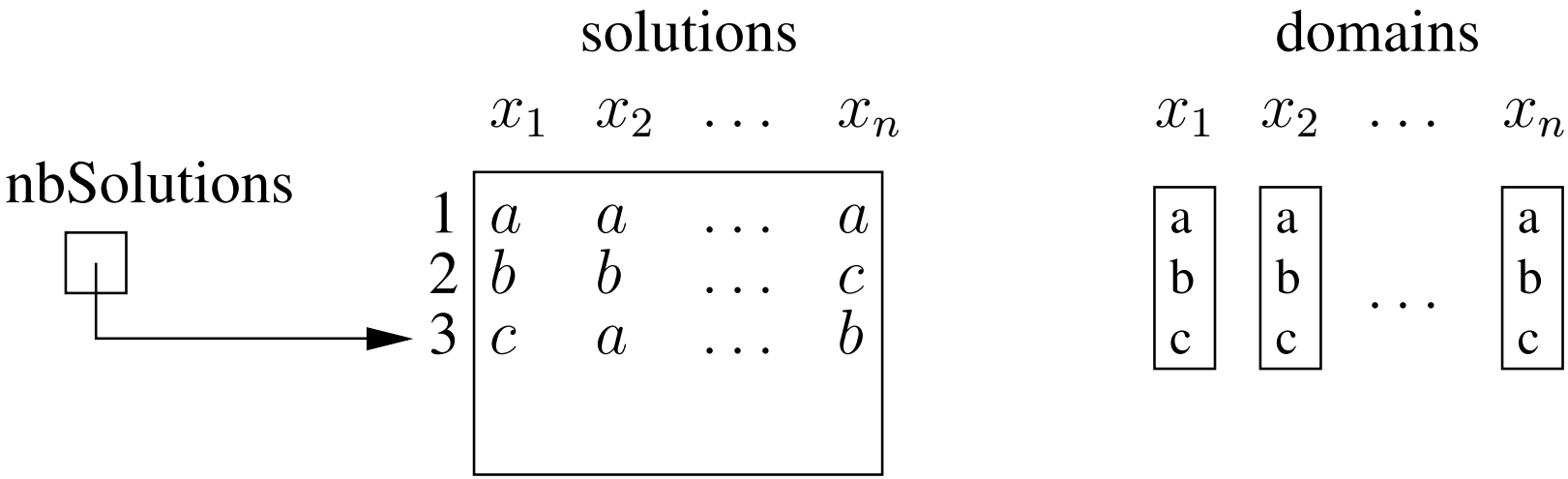
foreach *variable* $x \in S^{val}$ **do**
    if $I[x] \notin dom(x)$ **then**
        return *false*
return *true*

GIC4 - part 3/3

Algorithm 6: GIC4(P: CN)

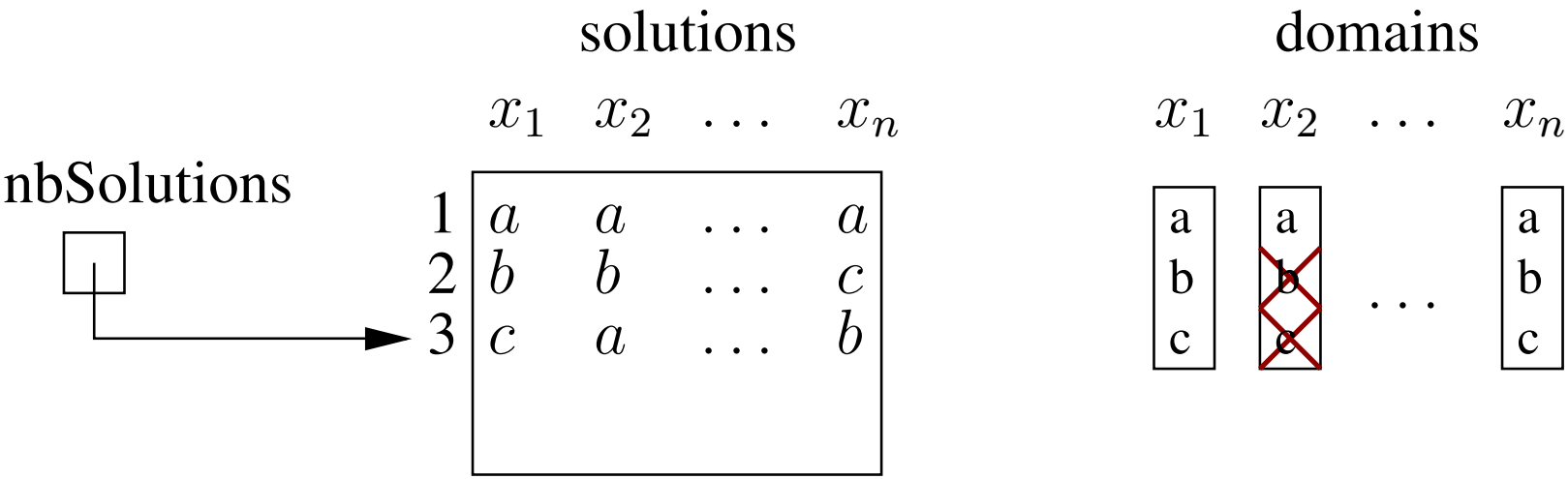
```
...
// Search for values not currently supported
foreach variable x ∈ S^sup do
    foreach value a ∈ dom(x) \ gicValues[x] do
        I ← searchSolutionFor(P|_{x=a})
        if I = nil then
            remove a from dom(x)
        else
            nbSolutions++
            solutions[nbSolutions] ← I
            handleSolution(I)
foreach variable x ∈ vars^fut(P) do
    lastSize[x] ← |dom(x)|
```

Illustration de GIC4



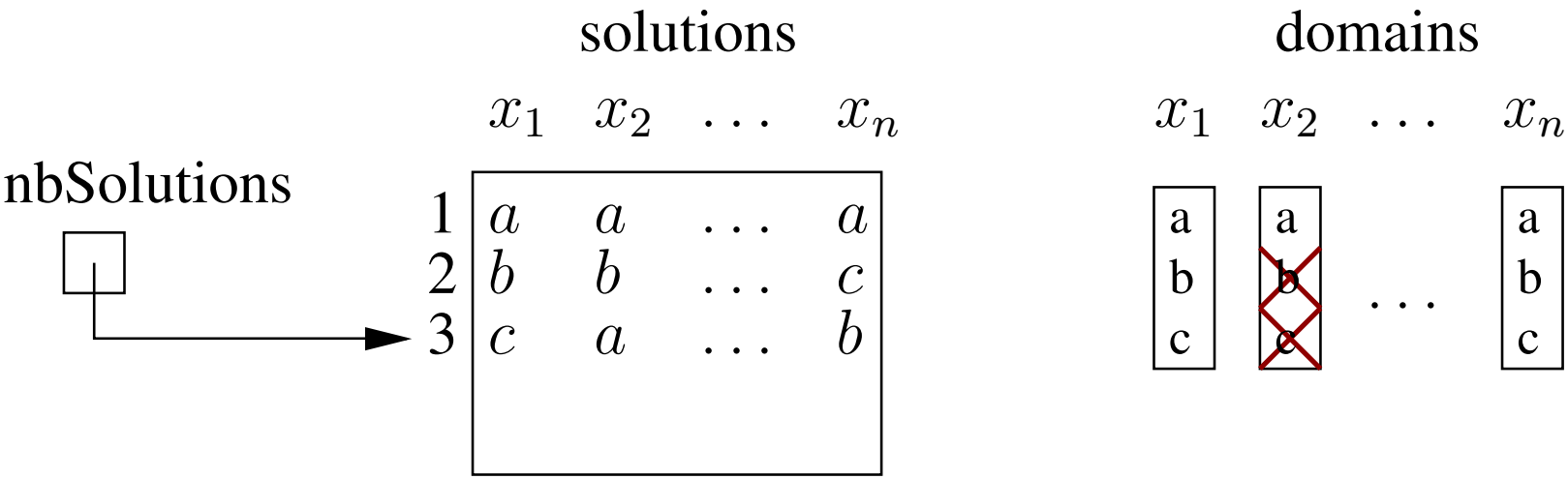
$\text{gicValues}[x_1]$:
$\text{gicValues}[x_2]$:
...
$\text{gicValues}[x_n]$:

Illustration de GIC4



$\text{gicValues}[x_1]$:
$\text{gicValues}[x_2]$:
...
$\text{gicValues}[x_n]$:

Illustration de GIC4



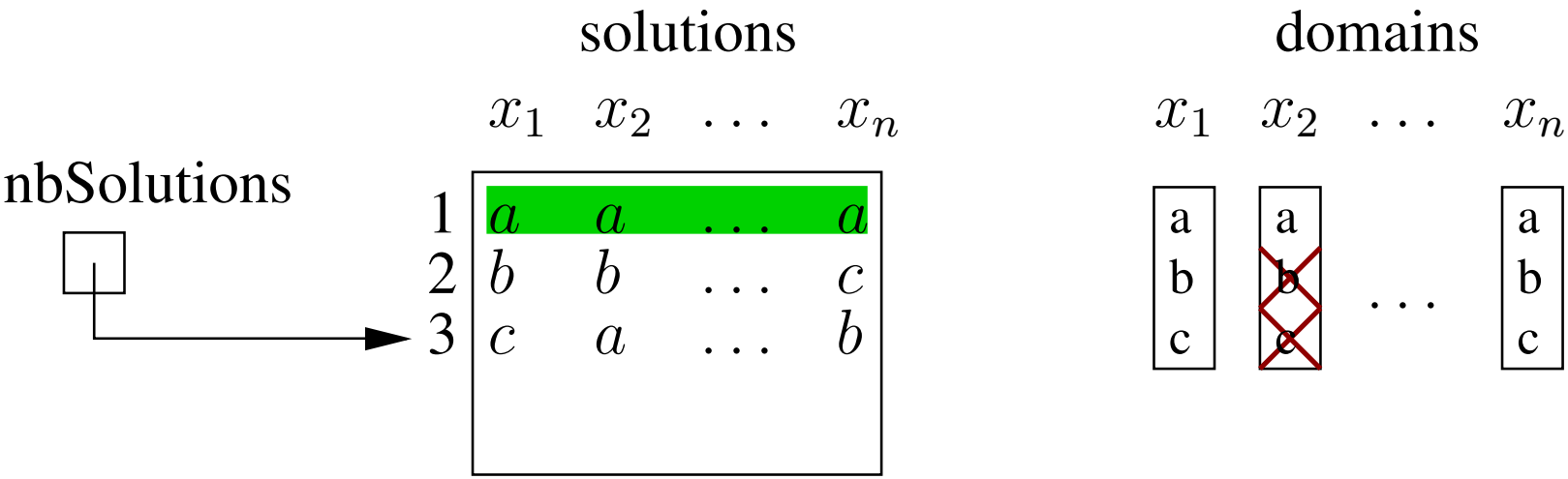
$\text{gicValues}[x_1] : \emptyset$

$\text{gicValues}[x_2] :$

...

$\text{gicValues}[x_n] : \emptyset$

Illustration de GIC4



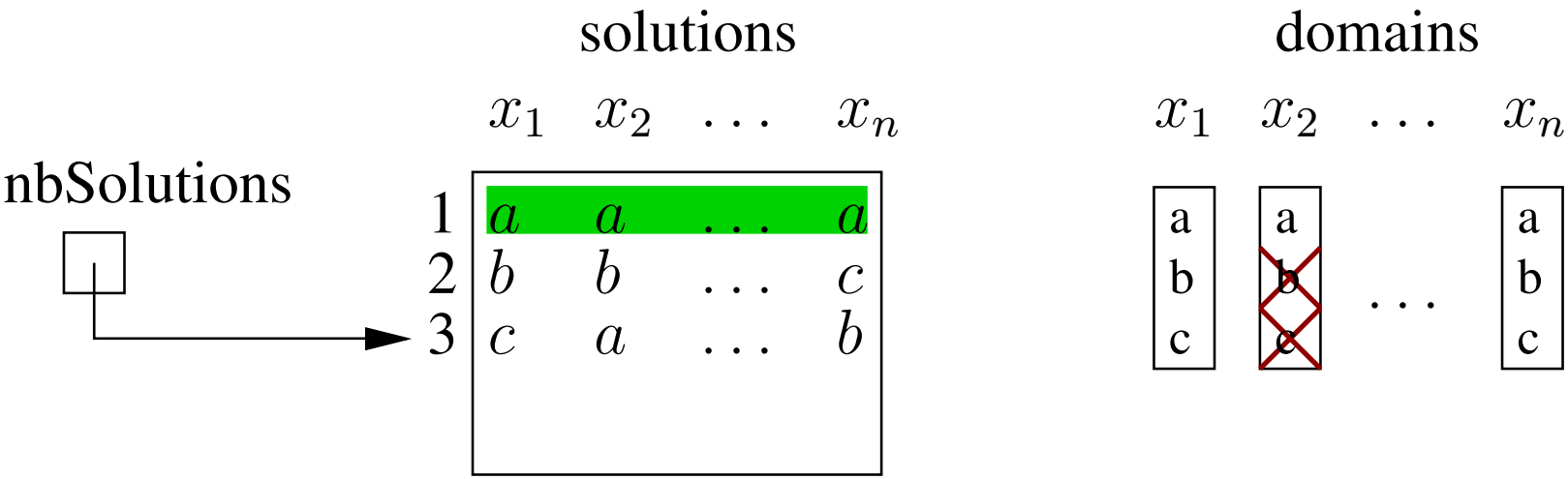
$\text{gicValues}[x_1] : \emptyset$

$\text{gicValues}[x_2] :$

...

$\text{gicValues}[x_n] : \emptyset$

Illustration de GIC4



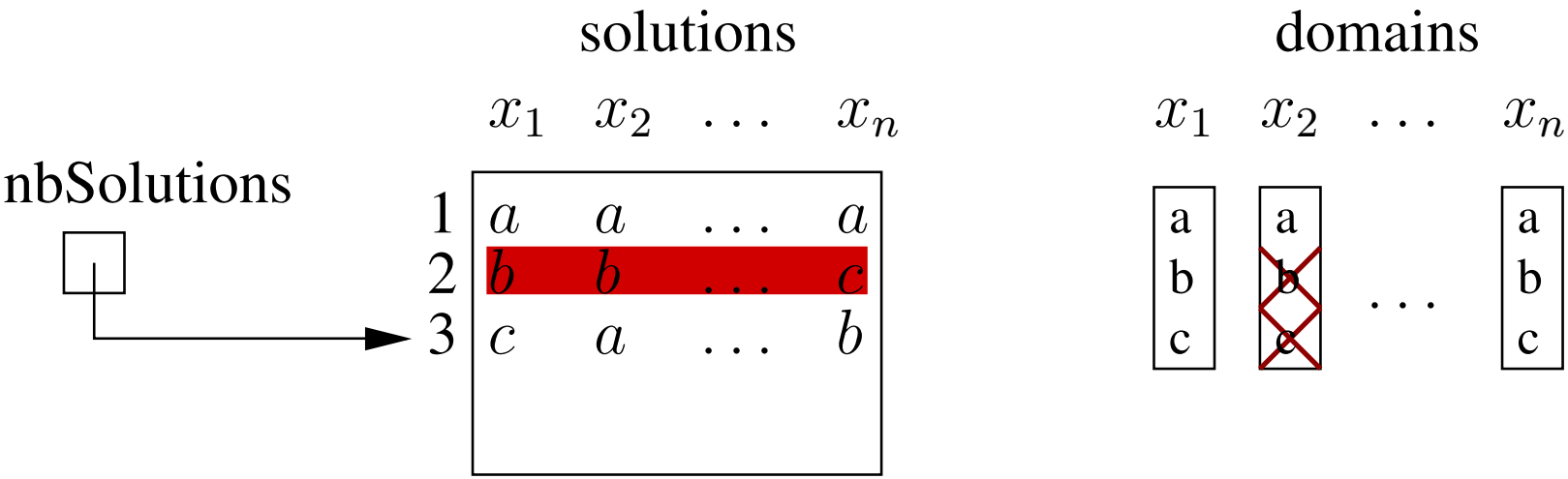
$\text{gicValues}[x_1] : \{a\}$

$\text{gicValues}[x_2] :$

...

$\text{gicValues}[x_n] : \{a\}$

Illustration de GIC4



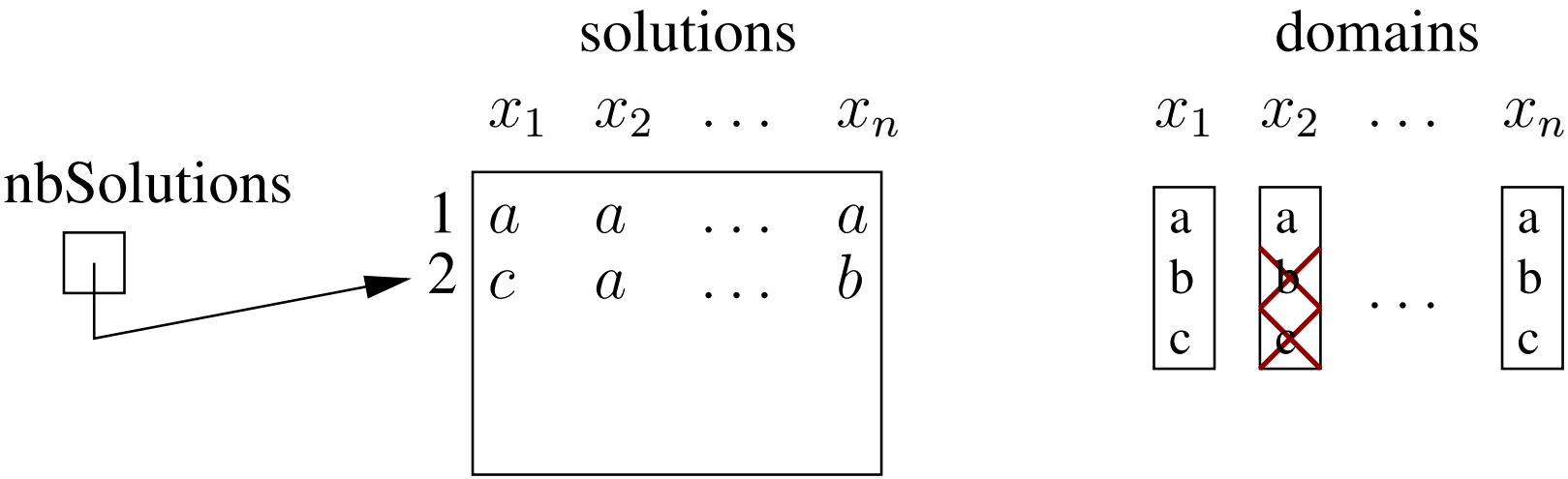
$\text{gicValues}[x_1] : \{a\}$

$\text{gicValues}[x_2] :$

...

$\text{gicValues}[x_n] : \{a\}$

Illustration de GIC4



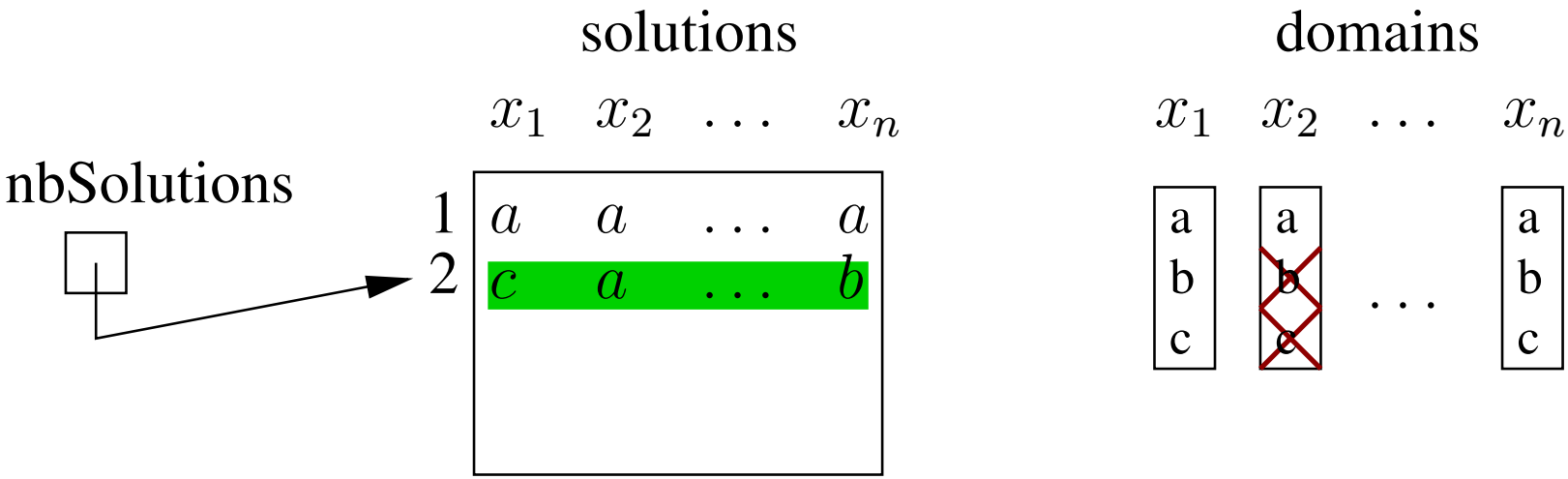
$\text{gicValues}[x_1] : \{a\}$

$\text{gicValues}[x_2] :$

...

$\text{gicValues}[x_n] : \{a\}$

Illustration de GIC4



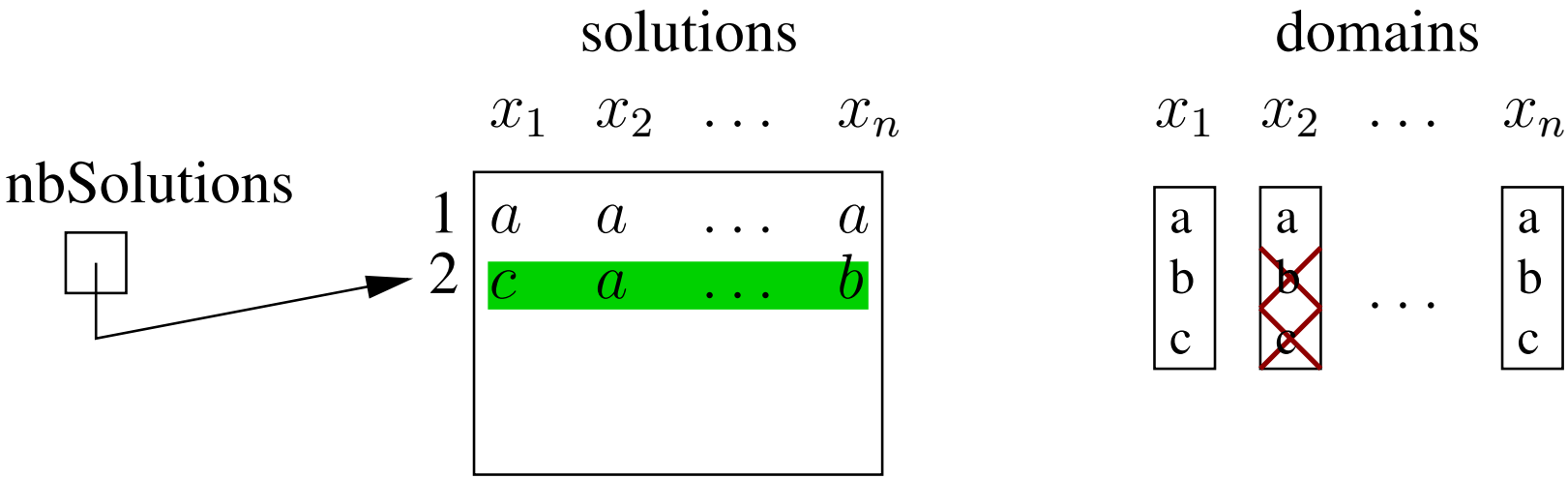
$\text{gicValues}[x_1] : \{a\}$

$\text{gicValues}[x_2] :$

...

$\text{gicValues}[x_n] : \{a\}$

Illustration de GIC4



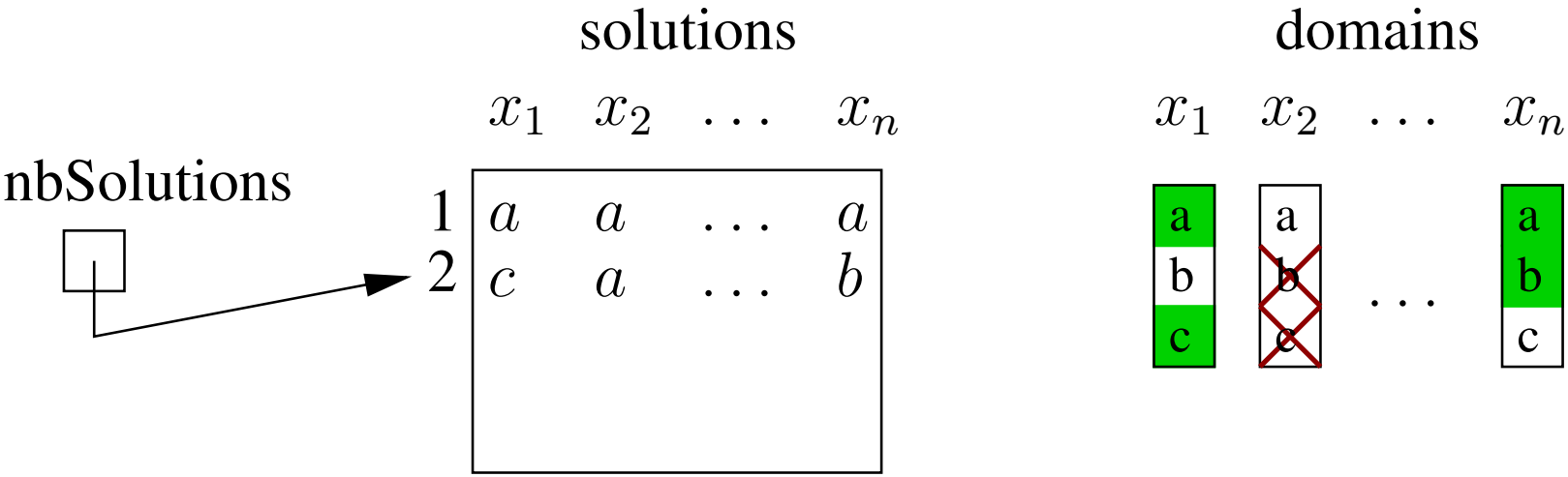
$\text{gicValues}[x_1] : \{a, c\}$

$\text{gicValues}[x_2] :$

...

$\text{gicValues}[x_n] : \{a, b\}$

Illustration de GIC4



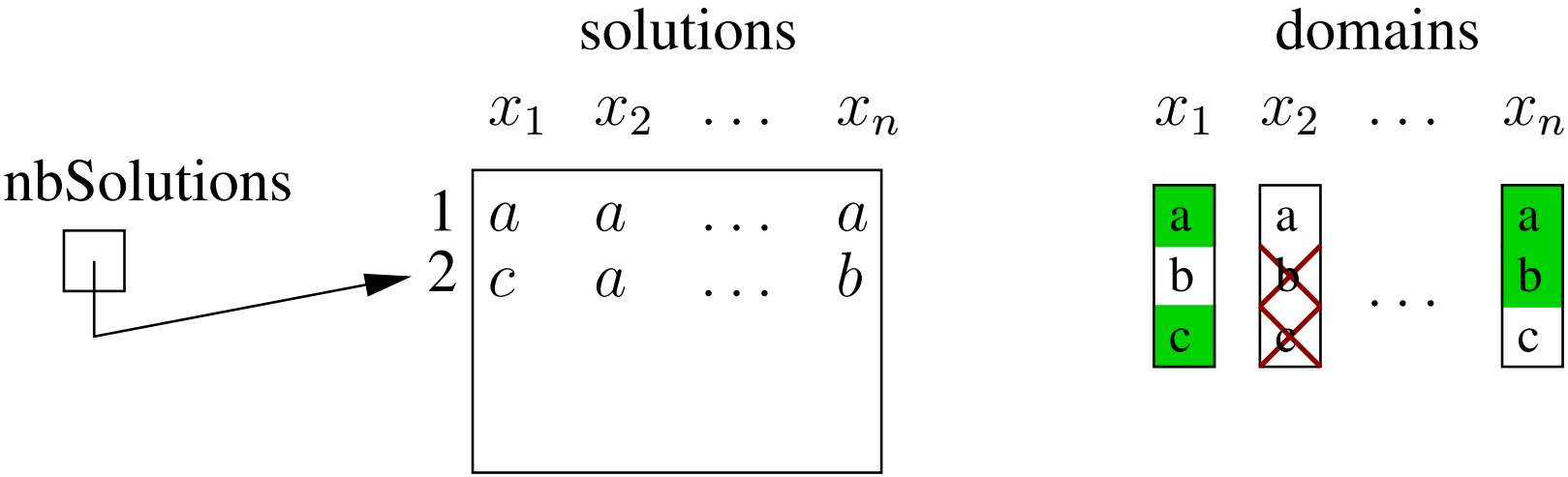
$\text{gicValues}[x_1] : \{a, c\}$

$\text{gicValues}[x_2] :$

...

$\text{gicValues}[x_n] : \{a, b\}$

Illustration de GIC4



$\text{gicValues}[x_1] : \{a, c\}$

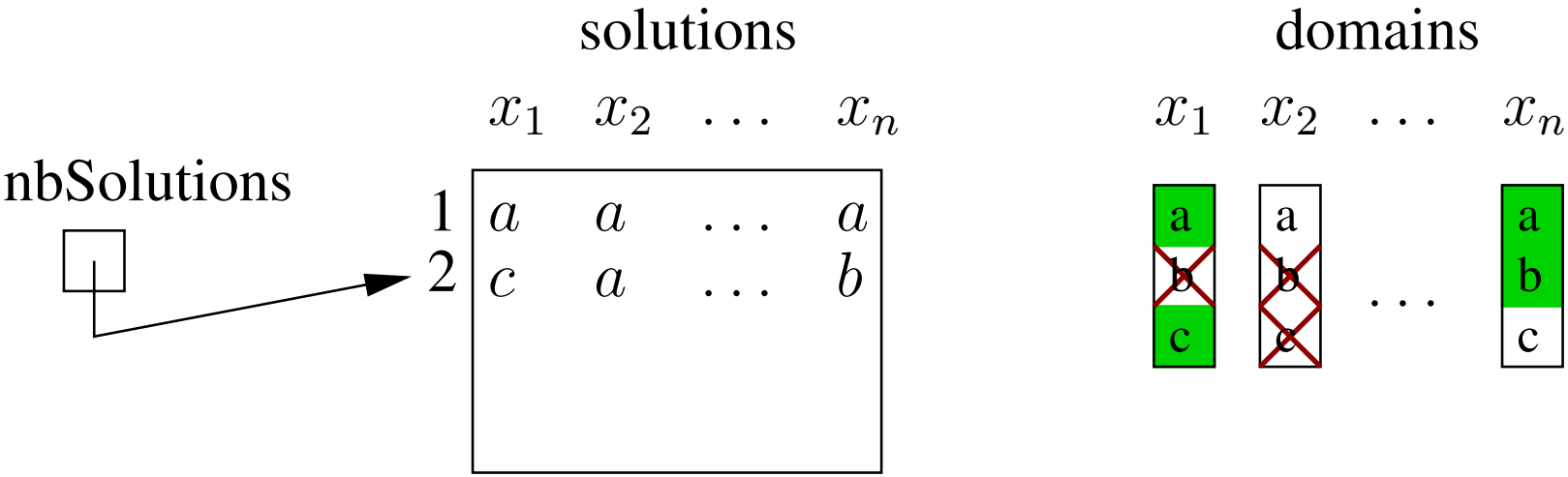
$\text{gicValues}[x_2] :$

...

$\text{gicValues}[x_n] : \{a, b\}$

searchSolutionFor $P|_{x_1=b}$?

Illustration de GIC4



$\text{gicValues}[x_1] : \{a, c\}$

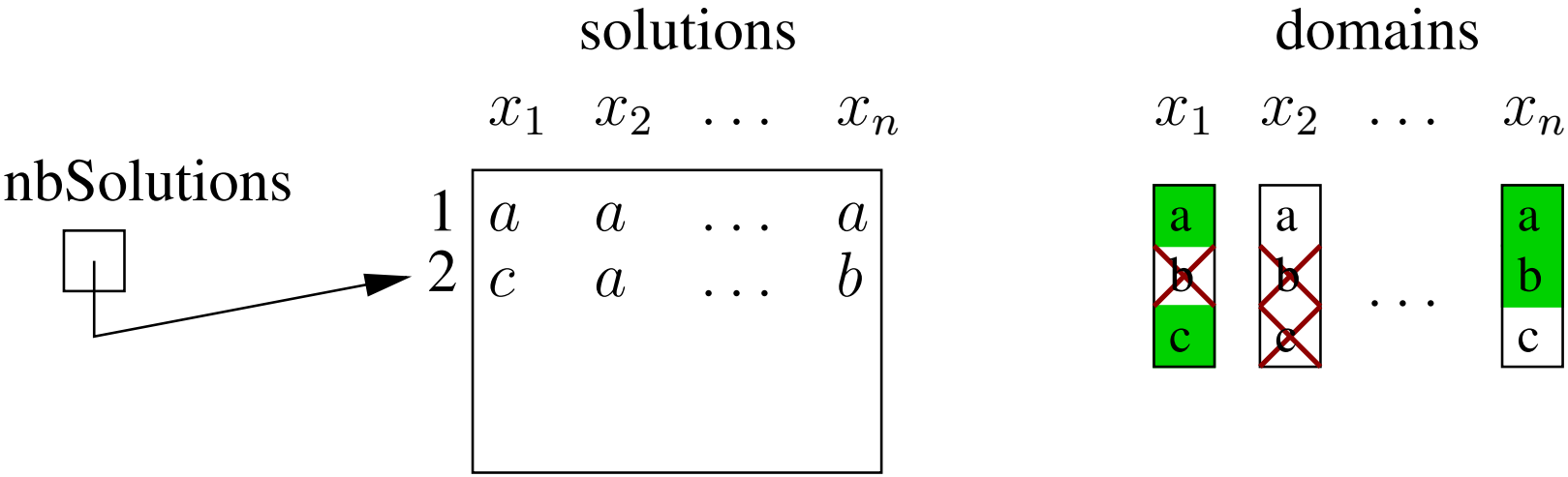
$\text{gicValues}[x_2] :$

...

$\text{gicValues}[x_n] : \{a, b\}$

searchSolutionFor $P|_{x_1=b}$?

Illustration de GIC4



$\text{gicValues}[x_1] : \{a, c\}$

$\text{gicValues}[x_2] :$

...

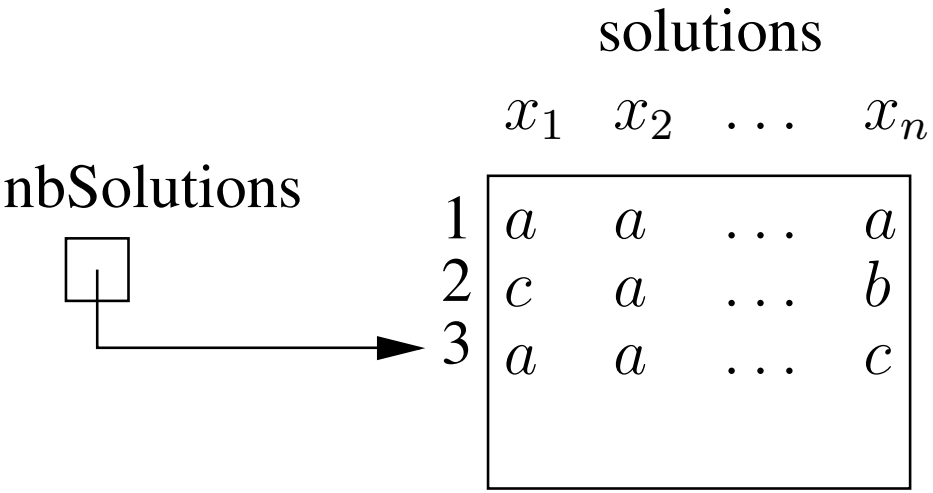
$\text{gicValues}[x_n] : \{a, b\}$

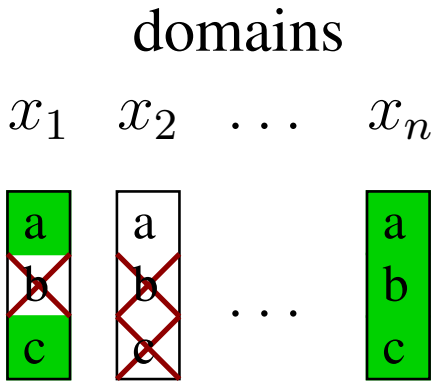
searchSolutionFor

$P|_{x_1=b}$?

$P|_{x_n=c}$?

Illustration de GIC4





gicValues$[x_1]$: $\{a, c\}$
gicValues$[x_2]$:
...
gicValues$[x_n]$: $\{a, b\}$

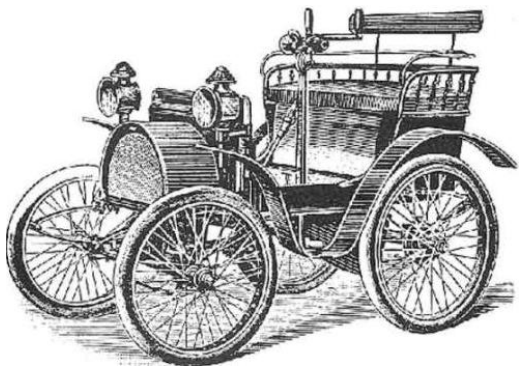
searchSolutionFor
$P|_{x_1=b}$? ✗
$P|_{x_n=c}$? ✓

Outline

Renault configuration instances

	n	d	e	r	t	D	T
souffleuse*	32	12	35	3	55	145	350
megane	99	42	113	10	48,721	396	194,838
master	158	324	195	12	26,911	732	183,701
small	139	16	147	8	222	340	3,044
medium	148	20	174	10	2,718	424	9,532
big	268	324	332	12	26,881	1,273	225,989

Table: Features of (Renault) configuration instances.



Maintaining GIC

	Establishing GIC with				Maintaining GIC with			
	GIC1	GIC2	GIC3	GIC4	GIC1	GIC2	GIC3	GIC4
souffleuse	0.02	0.01	0.01	0.01	0.13	0.07	**0.02**	**0.02**
megane	2.94	0.71	0.72	0.71	4.26	1.18	0.05	**0.04**
master	2.45	1.35	1.33	1.33	9.81	3.57	0.07	**0.06**
small	0.14	0.02	0.03	0.03	0.32	0.05	**0.01**	**0.01**
medium	0.26	0.04	0.05	0.04	0.35	0.04	**0.01**	**0.01**
big	4.19	1.16	1.10	1.10	12.6	2.60	**0.05**	**0.05**

Table: CPU time (in seconds) to establish GIC on Renault configuration instances, and to maintain it (average on 100 random greedy executions).

Number of conflicts

	souffleuse	megane	master
nFC2	252,605	313,910	time-out
MAC	0	7	5
	small	medium	big
nFC2	3,728	7,824	time-out
MAC	0	3	3

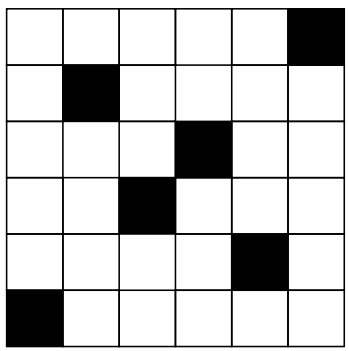
Table: Number of conflicts encountered when running nFC2 and MAC (sum over 100 random executions).

Remember: with GIC maintained, no conflict is possible

Crosswords instances

	Establishing GIC with				Maintaining GIC with			
	GIC1	GIC2	GIC3	GIC4	GIC1	GIC2	GIC3	GIC4
ogd-vg5-5	2.25	0.67	0.67	0.67	2.34	0.79	0.73	**0.70**
ogd-vg5-6	6.40	2.18	2.19	2.19	7.42	2.82	2.58	**2.48**
ogd-vg5-7	25.8	9.91	9.87	9.84	33.4	15.2	14.3	**13.8**

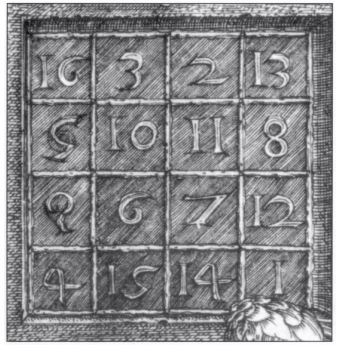
Table: CPU time (in seconds) to establish GIC on some Crosswords instances, and to maintain it on average on 100 random greedy executions.



Puzzle instances (finding a one-solution state)

	Establishing GIC with				Maintaining GIC with			
	GIC1	GIC2	GIC3	GIC4	GIC1	GIC2	GIC3	GIC4
sudoku-9x9	1.58	0.32	0.32	0.31	15.3	2.71	2.10	**1.74**
sudoku-16	6.04	0.51	0.50	0.50	246	25.5	26.5	**18.9**
magic-4x4	0.96	0.26	0.28	0.28	1.63	**0.69**	0.71	0.71
magic-5x5	14.7	3.01	3.10	2.99	55.1	15.9	15.6	**13.7**

Table: CPU time (in seconds) on average on 100 random greedy executions until a unique solution is found.



Positive Consistency

Definition

A constraint c is positively consistent in a CN P iff any valid tuple $\tau \in rel(c)$ can be extended to a solution of P.

	souffleuse	megane	master
CPU	0.68	352	368
# tuples removed	0	138,493	90,874

	small	medium	big
CPU	2.6	4.2	613
# tuples removed	240	5,425	105,020

Table: CPU time (in seconds) and number of tuples deleted when establishing positive consistency on (Renault) configuration instances.

Conclusion

The Graal for configuration: GIC

We have:

- presented some complexity results about GIC
- devised a few GIC algorithms
- shown that maintaining GIC was effective

Future work:

- GIC restored upon constraint retractions
- GIC for soft constraints