Domain-Heuristics for Arc-Consistency Algorithms

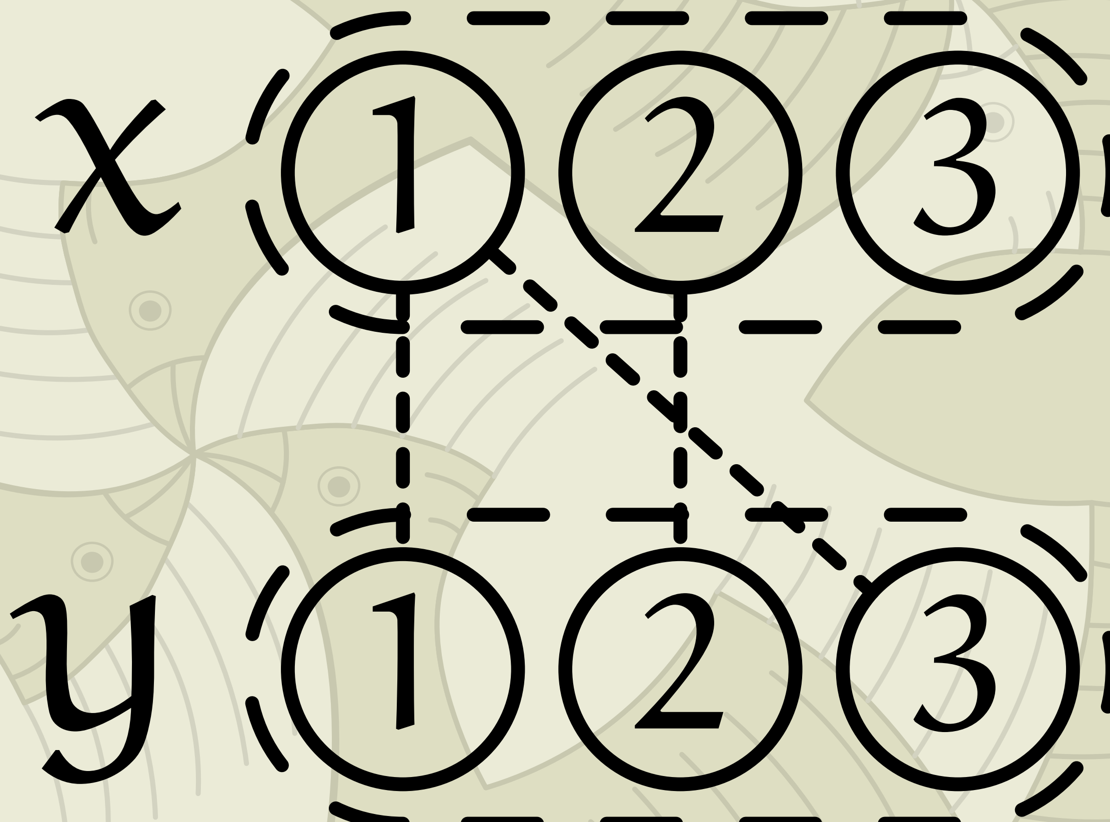
M.R.C. van Dongen
Compute Science Department UCC
Cork Constraint Computation Centre
Homepage: http://csweb.ucc.ie/~dongen.html
Email: dongen@cs.ucc.ie

June 19, 2002

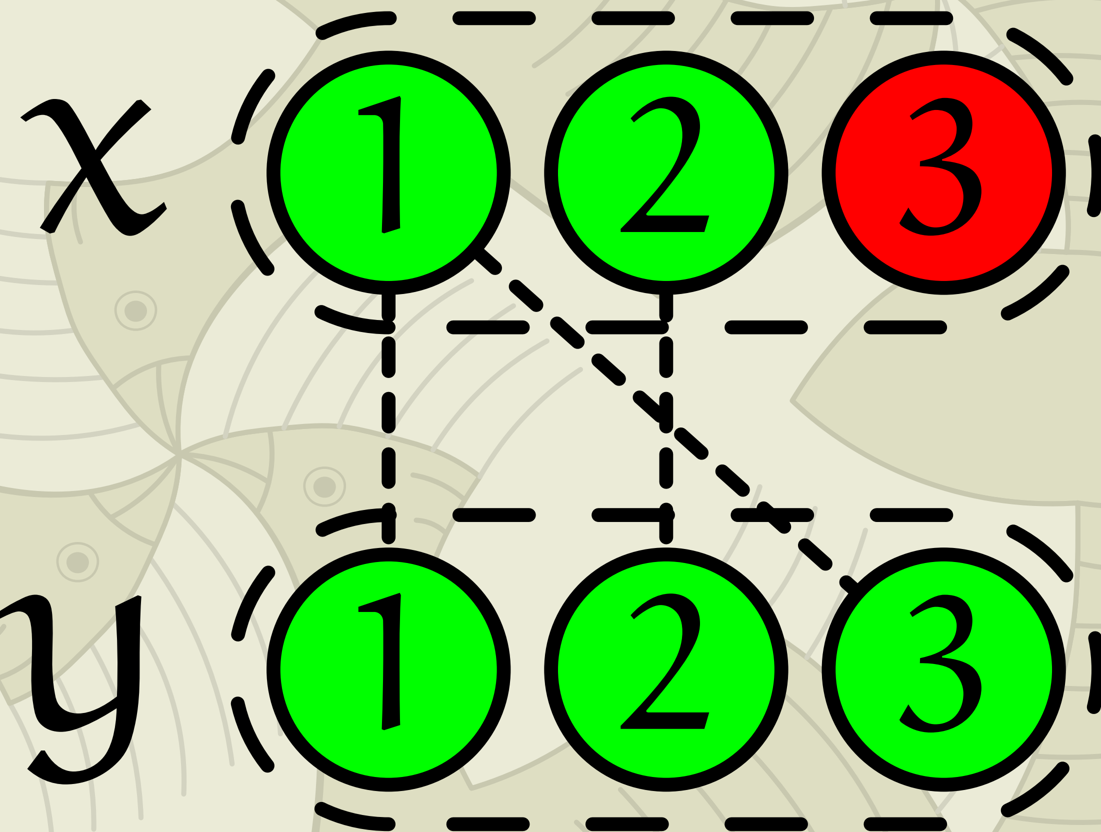
Outline

- Arc-Consistency;
- Case Study;
- Results;
- Discussion and Future Work.

Arc-Consistency



Arc-Consistency



Heuristics

Arc-consistency algorithms carry out *support-checks* to find out about the properties of CSPs.

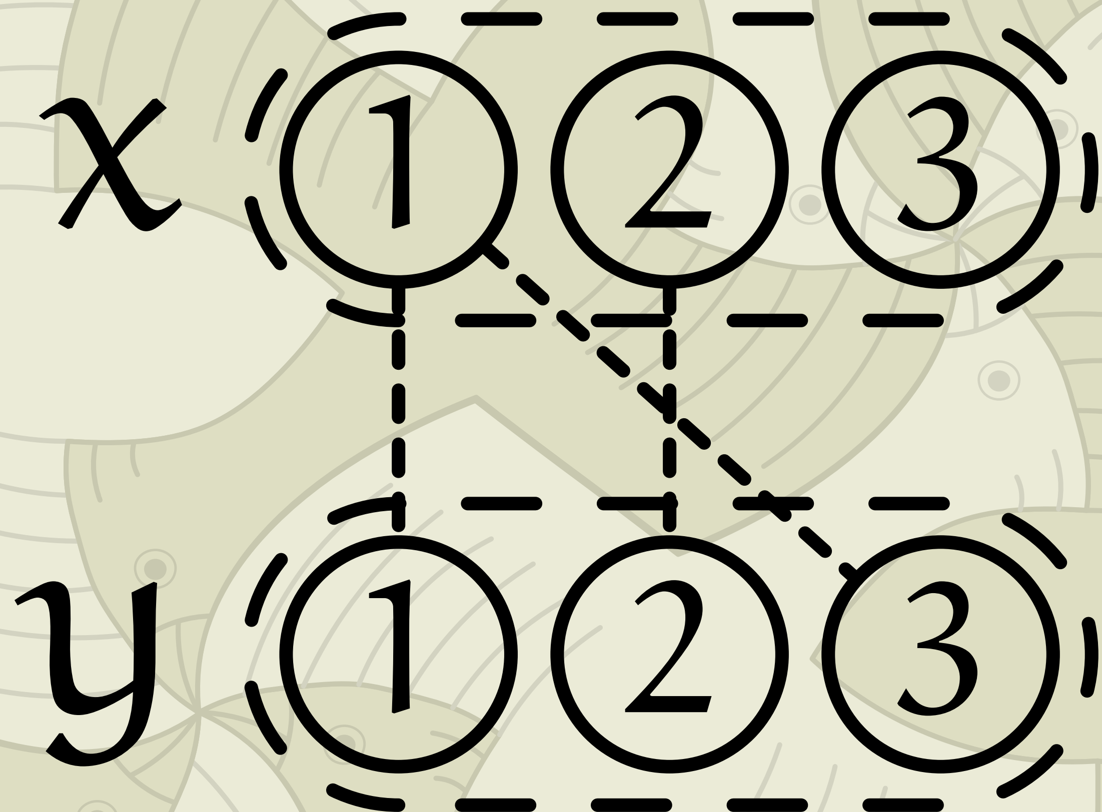
They use *arc-heuristics* to select the constraint that will be used for the next support-check.

They use *domain-heuristics* to select the values that will be used for the next support-check.

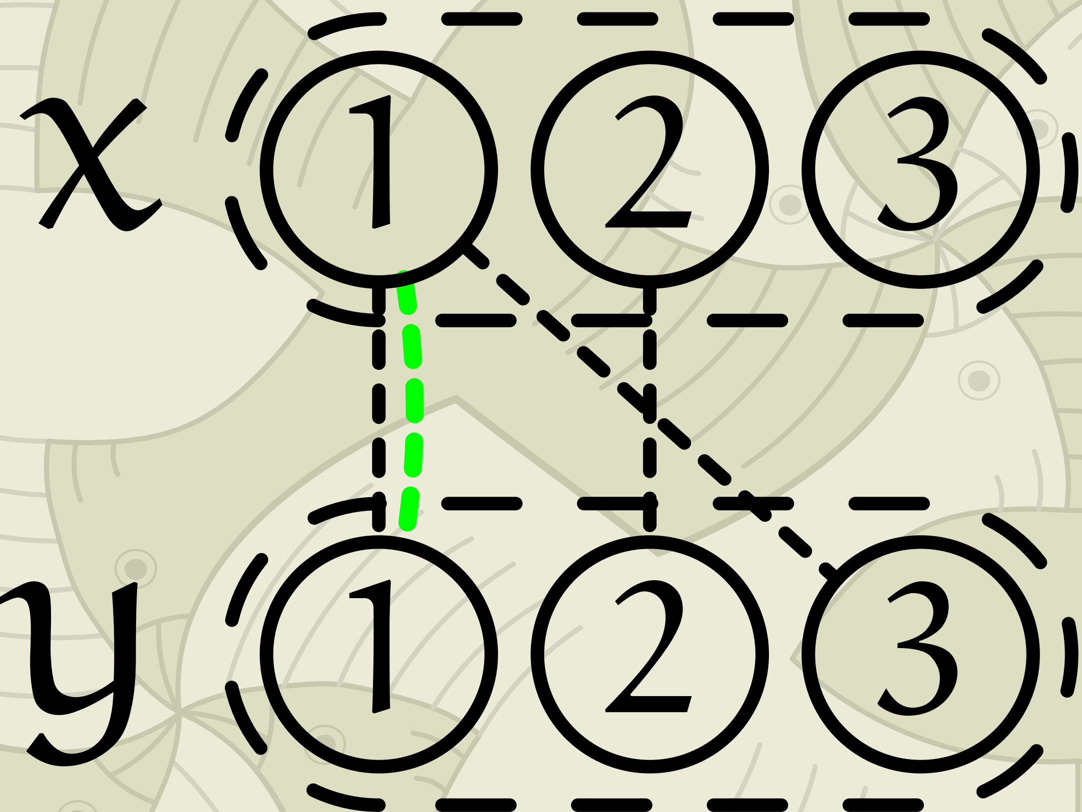
Domain-Heuristic $\mathcal{L}$

Arc-Consistency Algorithms come in many different flavours.

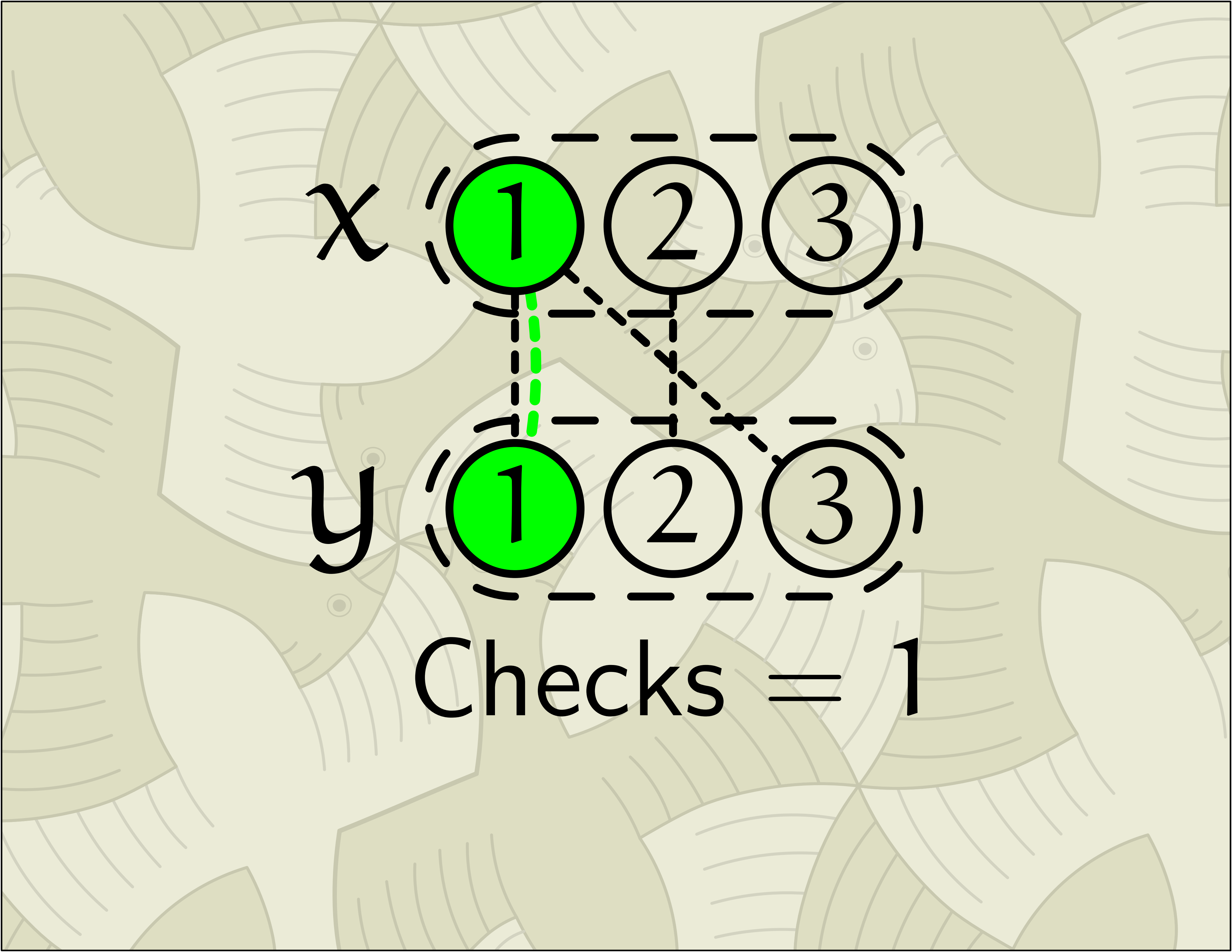
The current state-of-the-art is called AC-7. It never repeats support-checks and has a $\mathbf{O}(ed)$ space-complexity. AC-7 normally comes equipped with a lexicographical domain-heuristic $\mathcal{L}$.

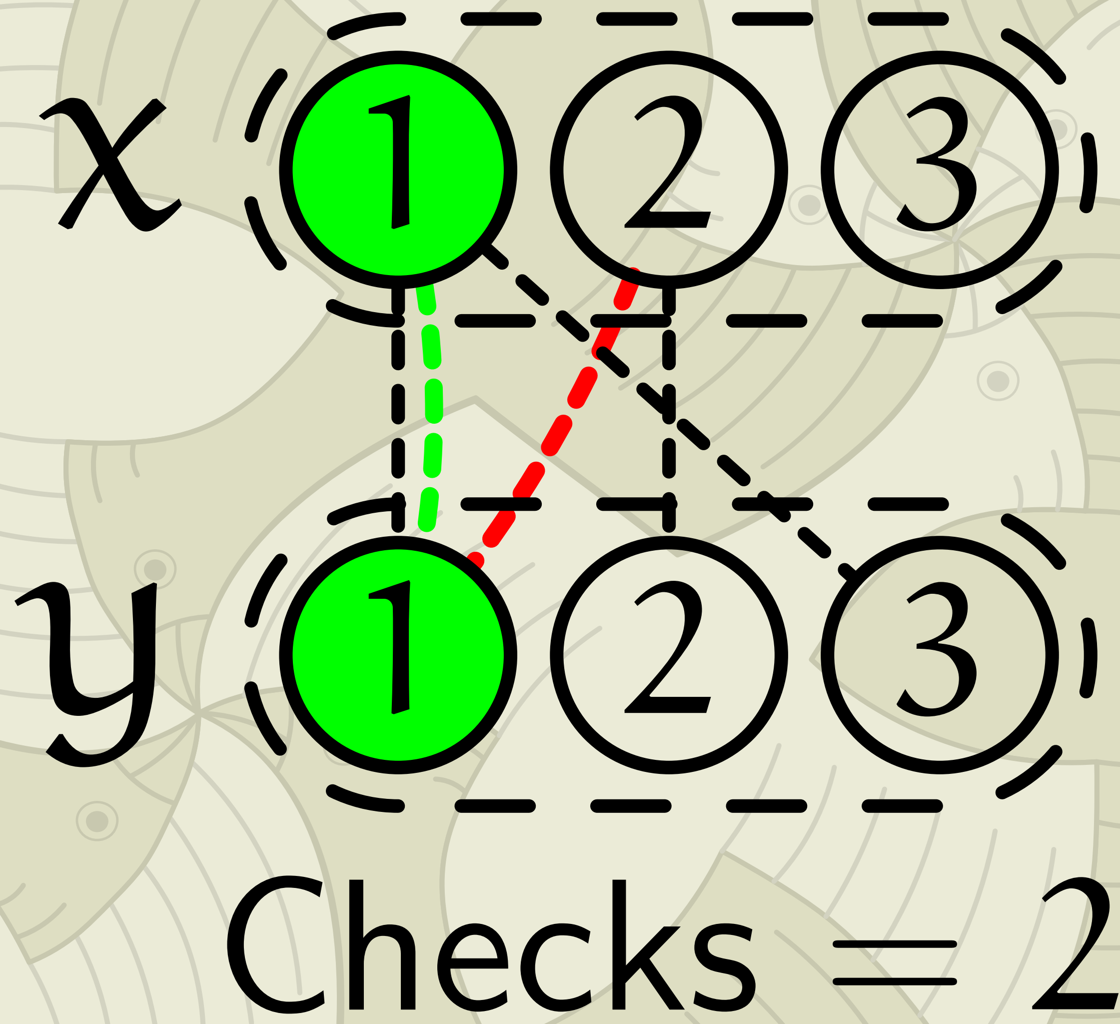


Checks = 0

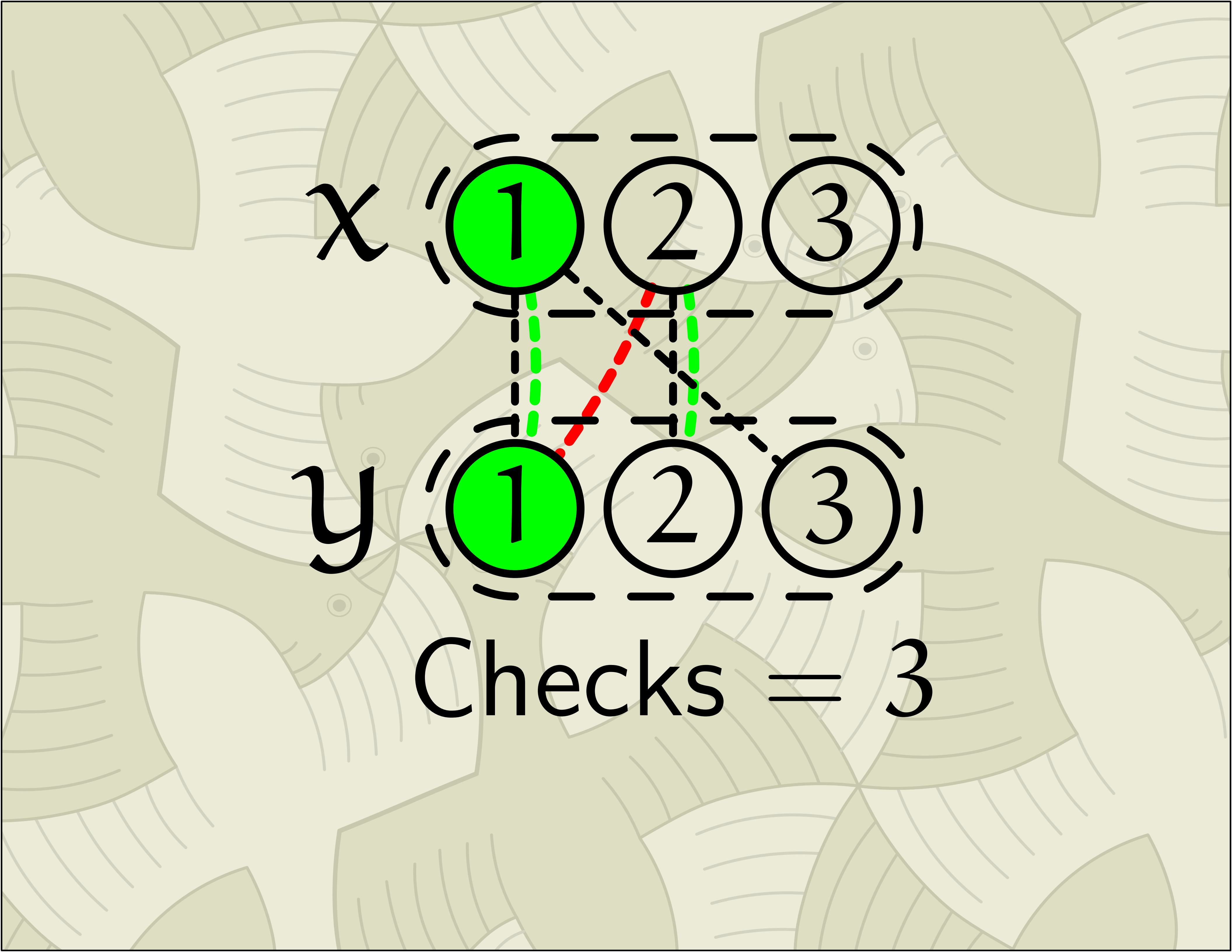


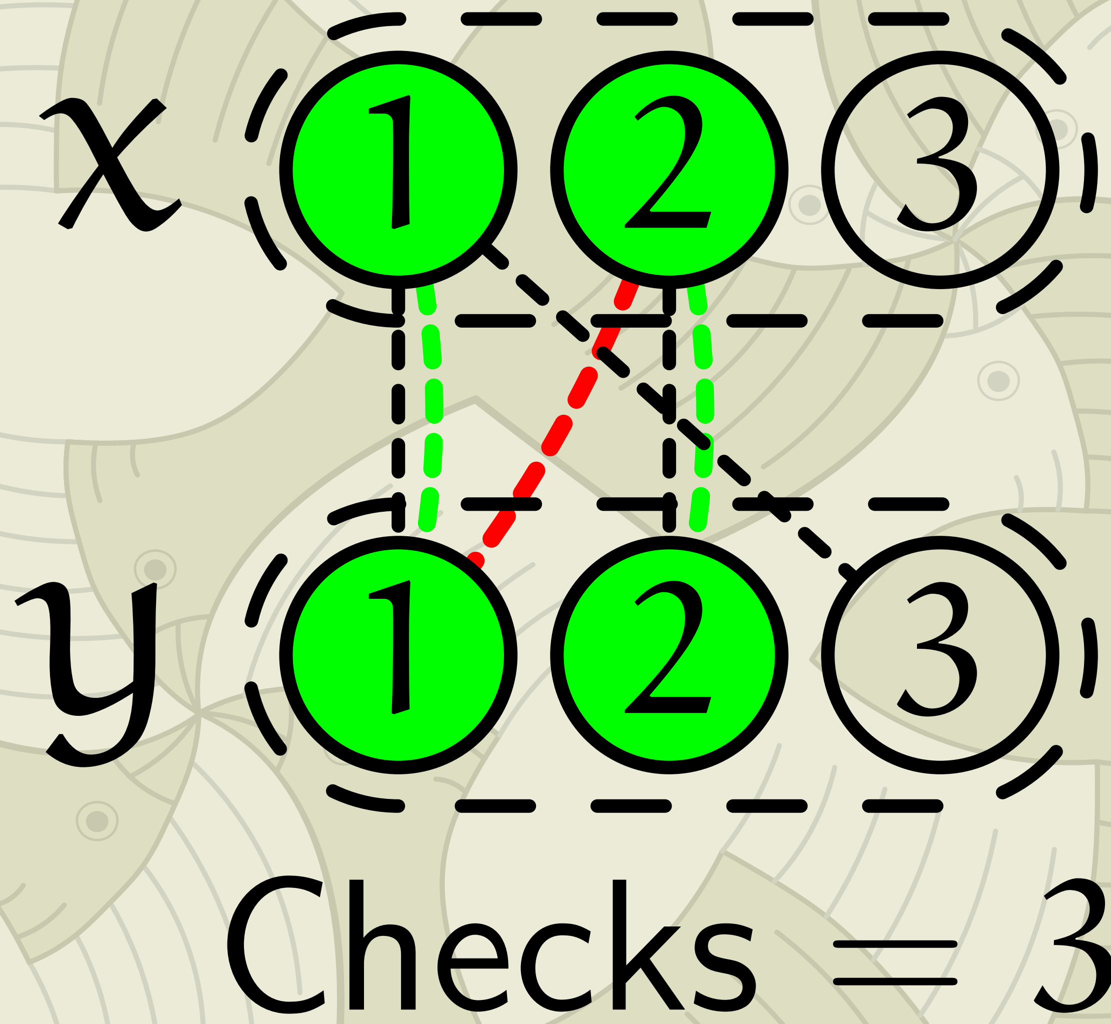
Checks = 1

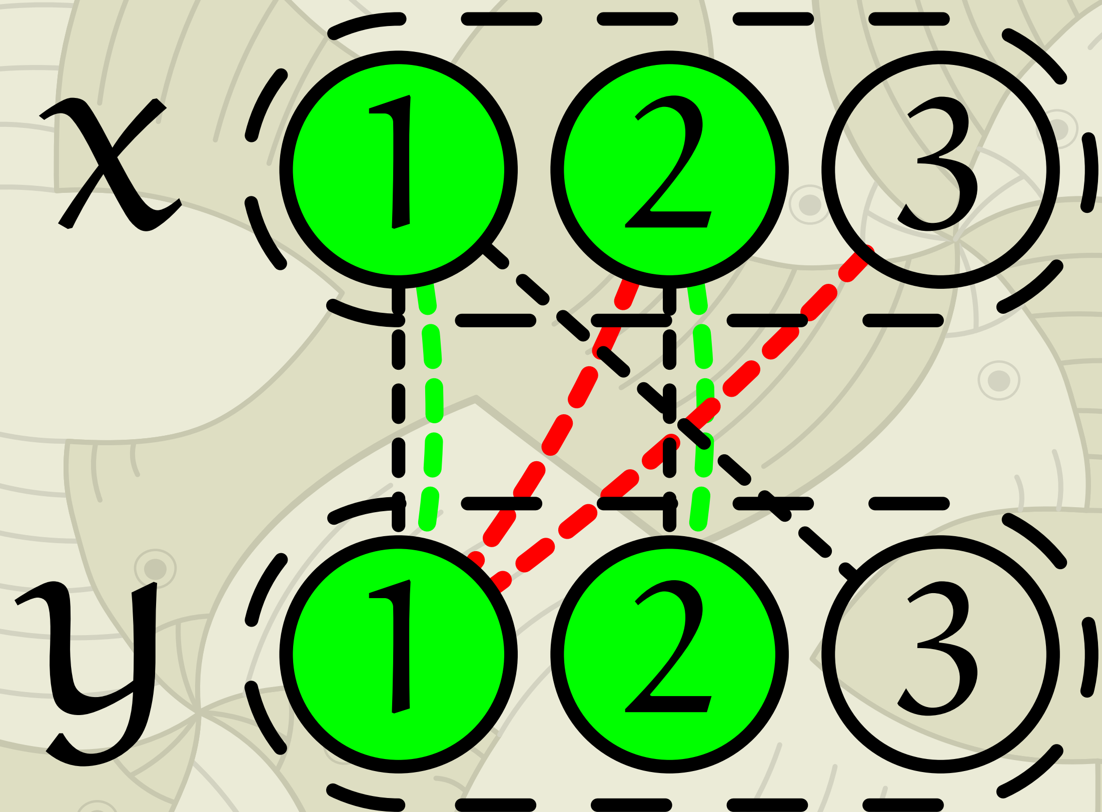

x
1
2
3
y
1
2
3
Checks = 1



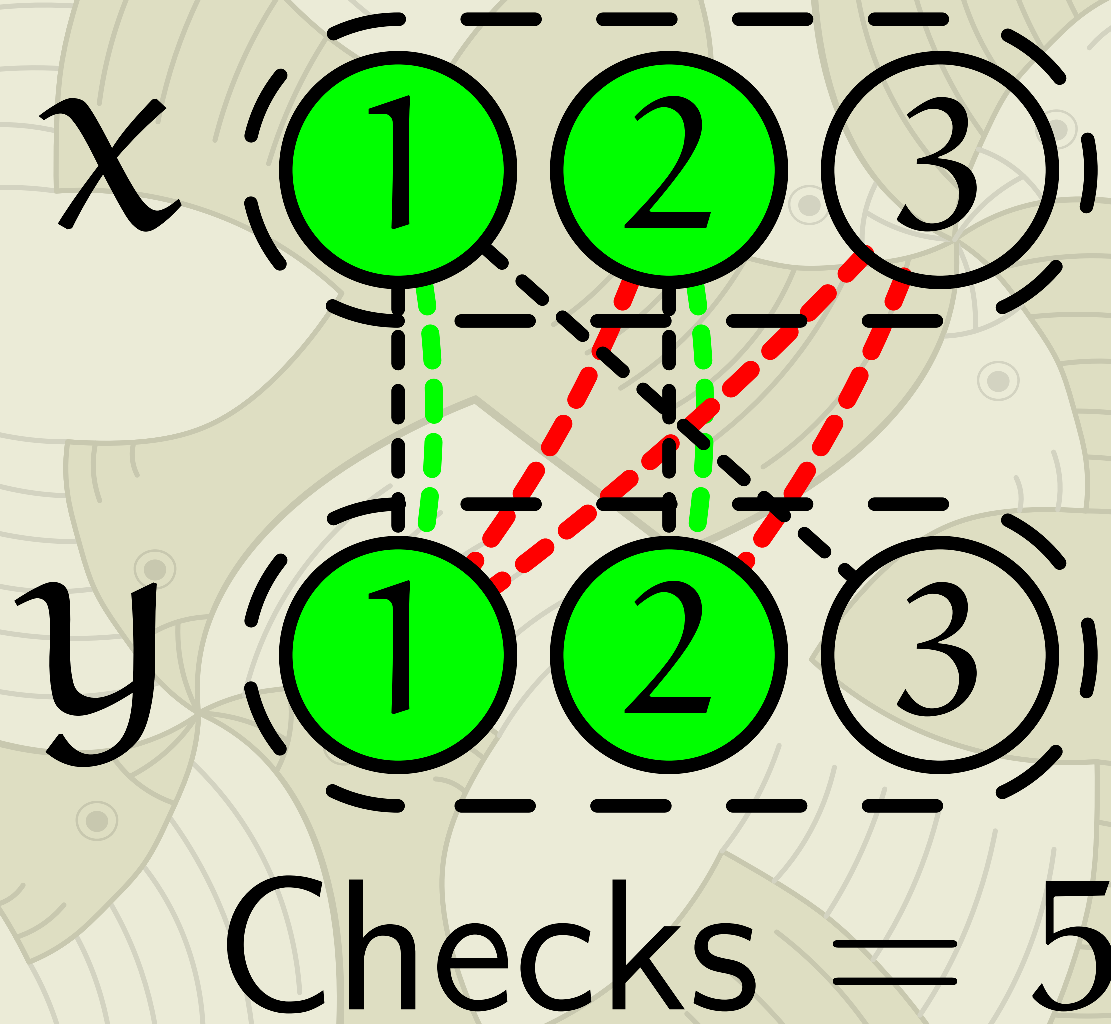
x
1
2
3
y
1
2
3
Checks = 2


x
1
2
3
y
1
2
3
Checks = 3

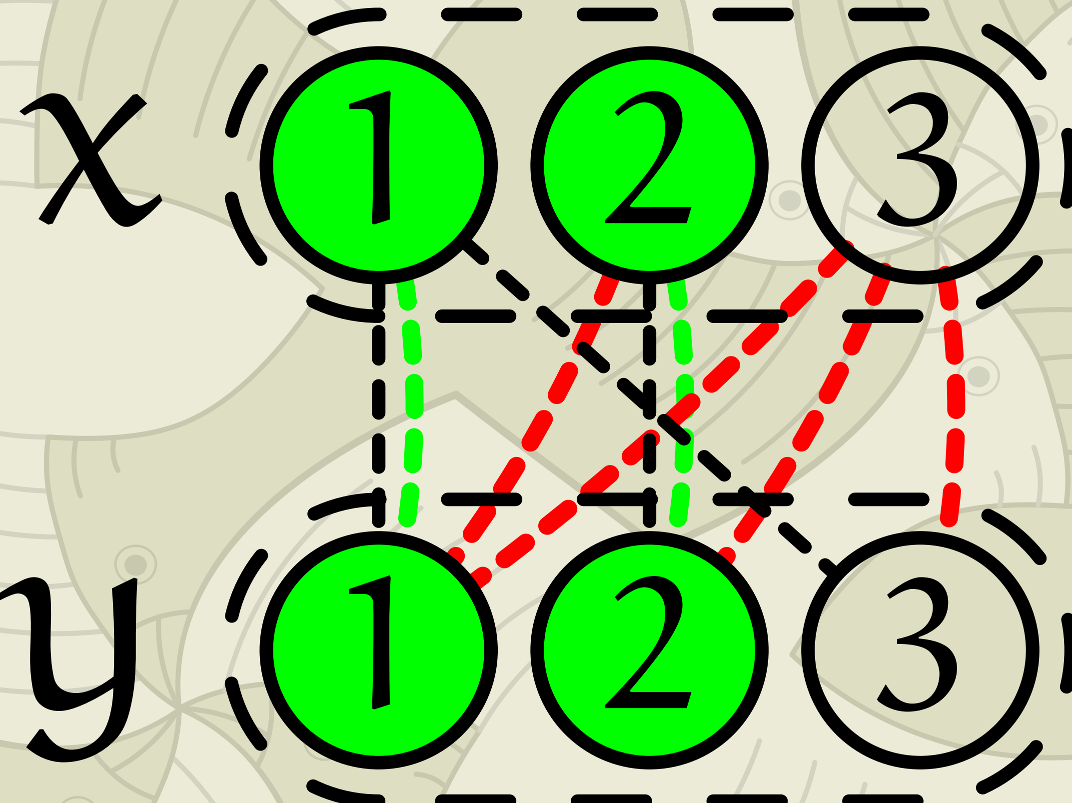

x
1
2
3
y
1
2
3
Checks = 3



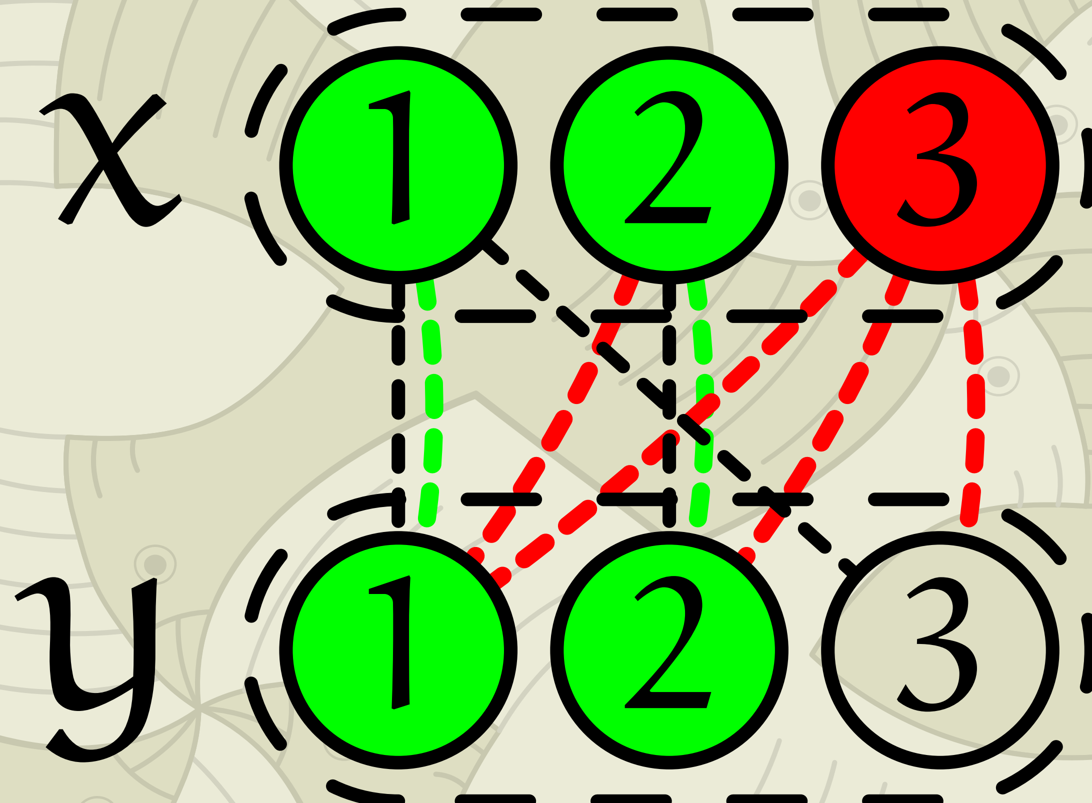
Checks = 4



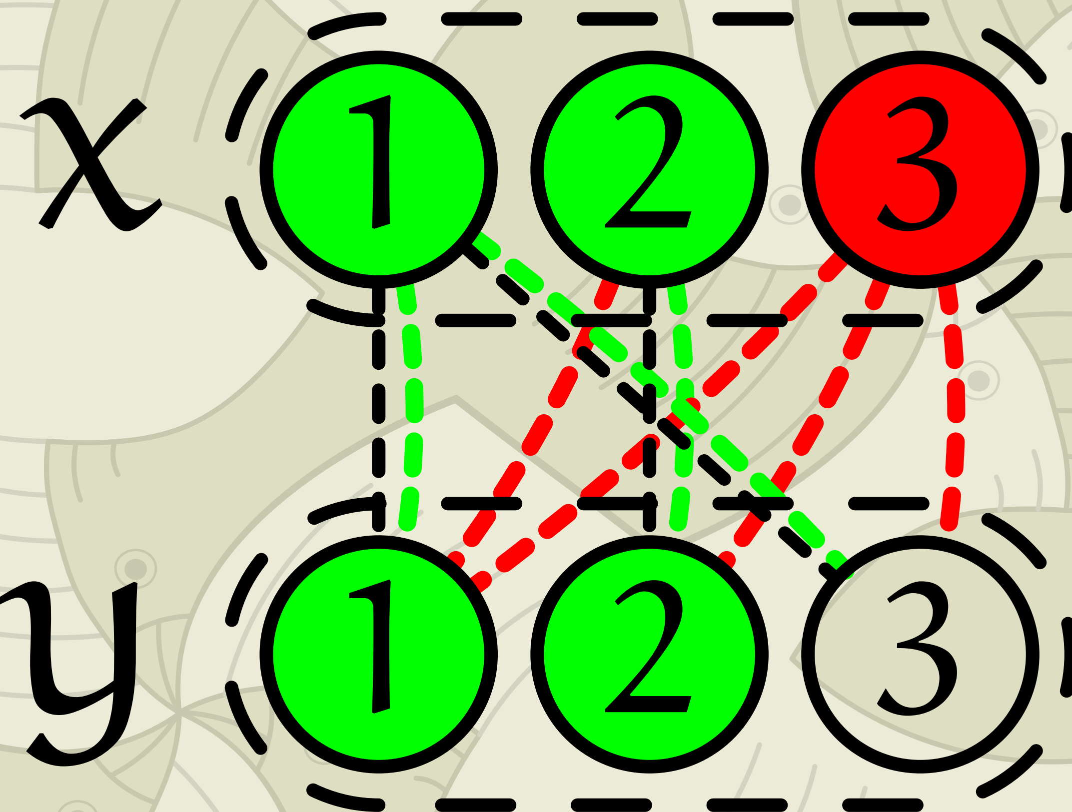
x
1
2
3
y
1
2
3
Checks = 5



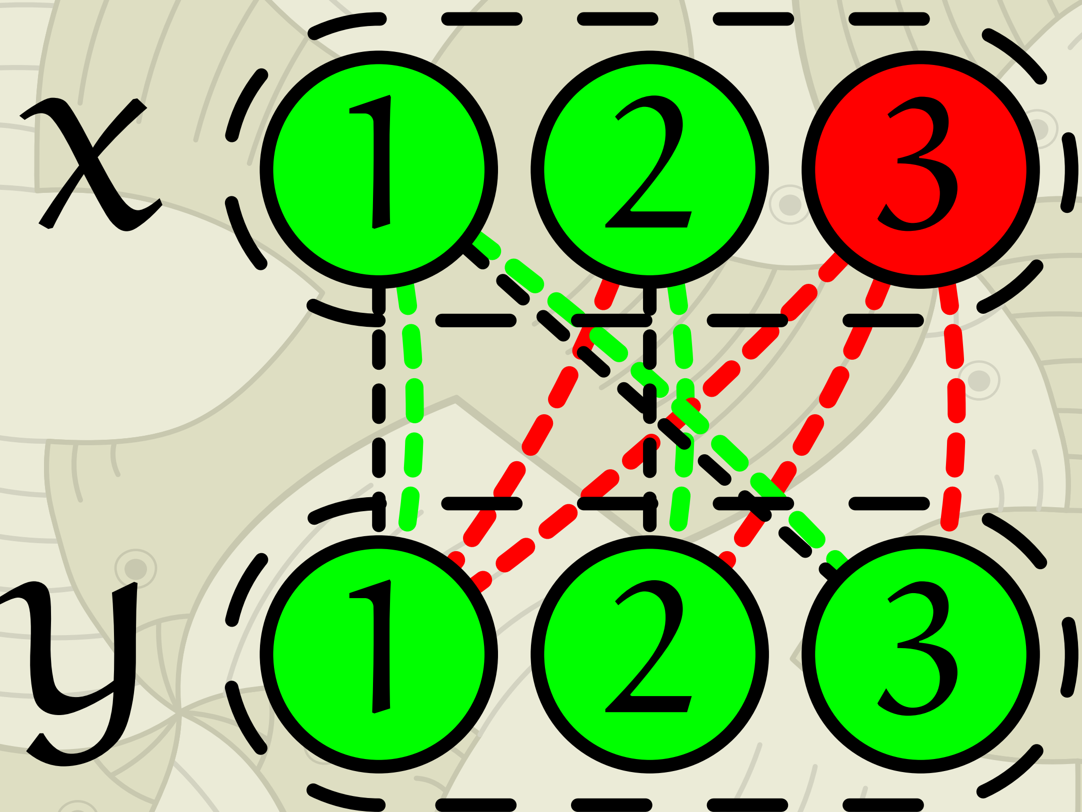
Checks = 6



Checks = 6



Checks = 7

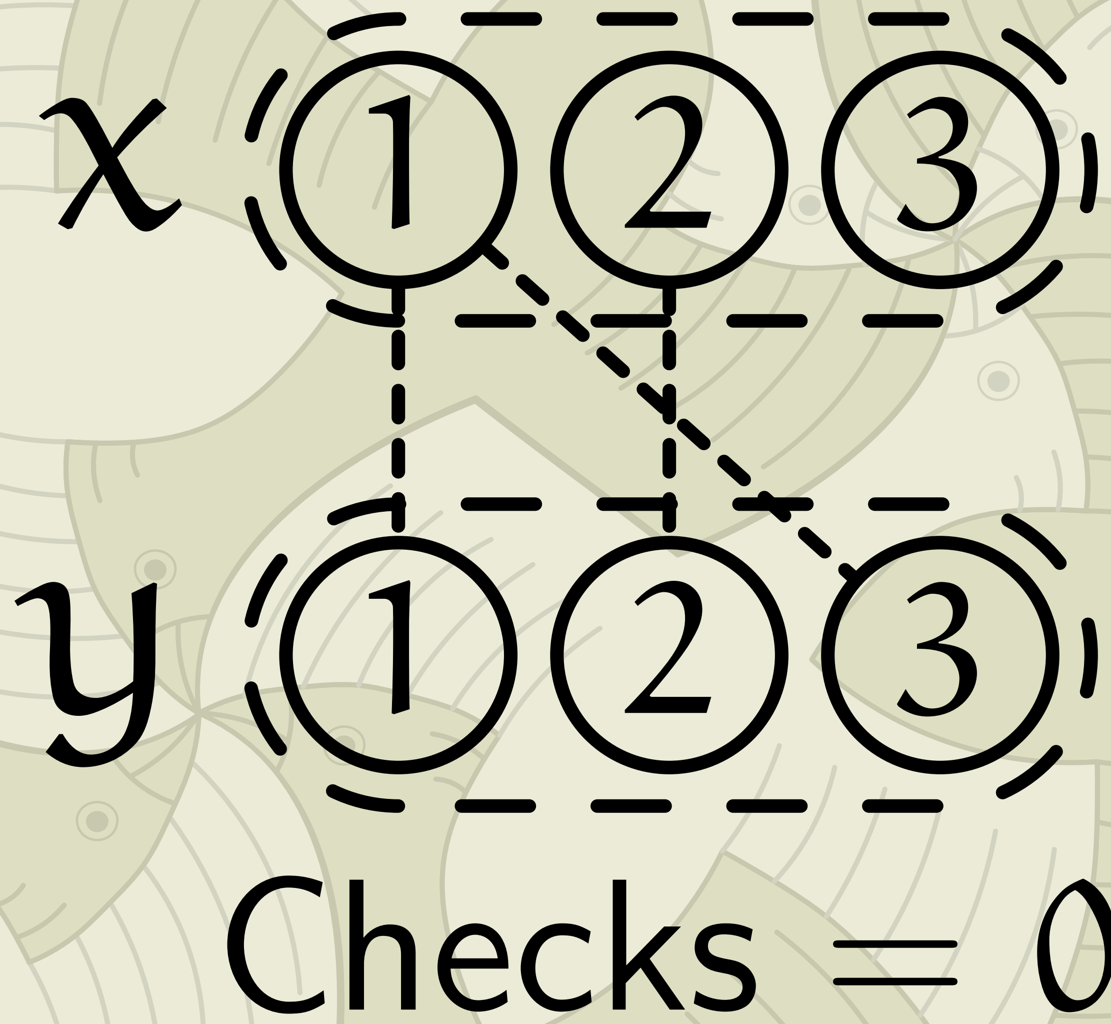


Checks = 7

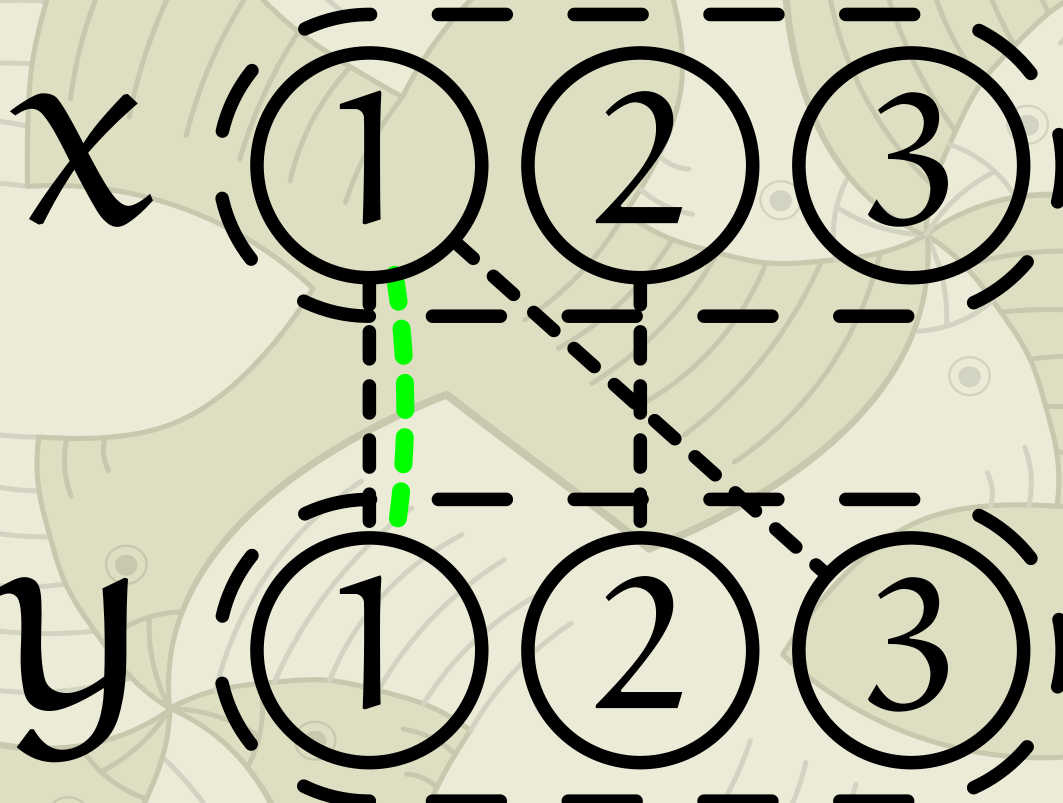
Domain-Heuristic $\mathcal{D}$

A heuristic which seeks to maximise the number of *double-support* checks.

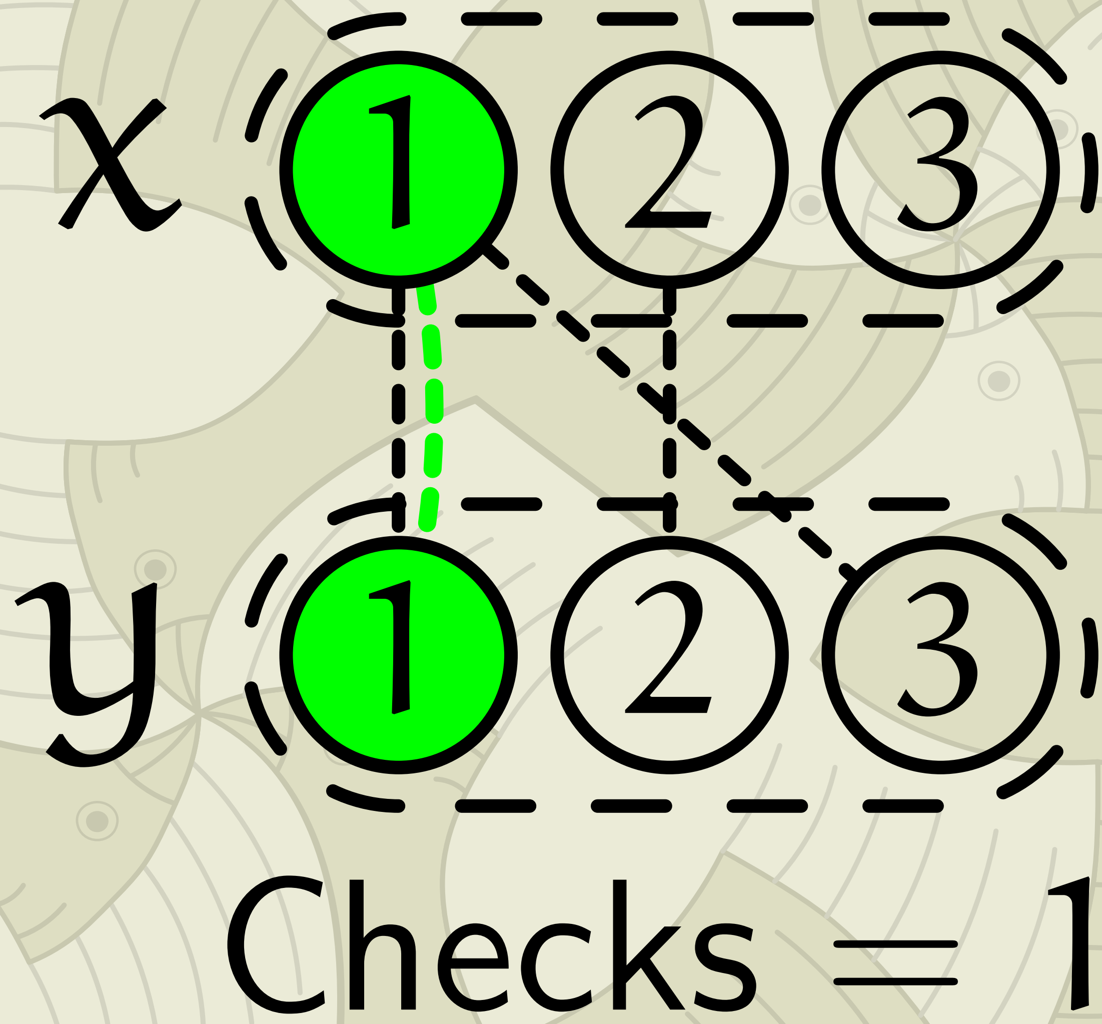
This heuristic can be incorporated into most arc-consistency algorithms.

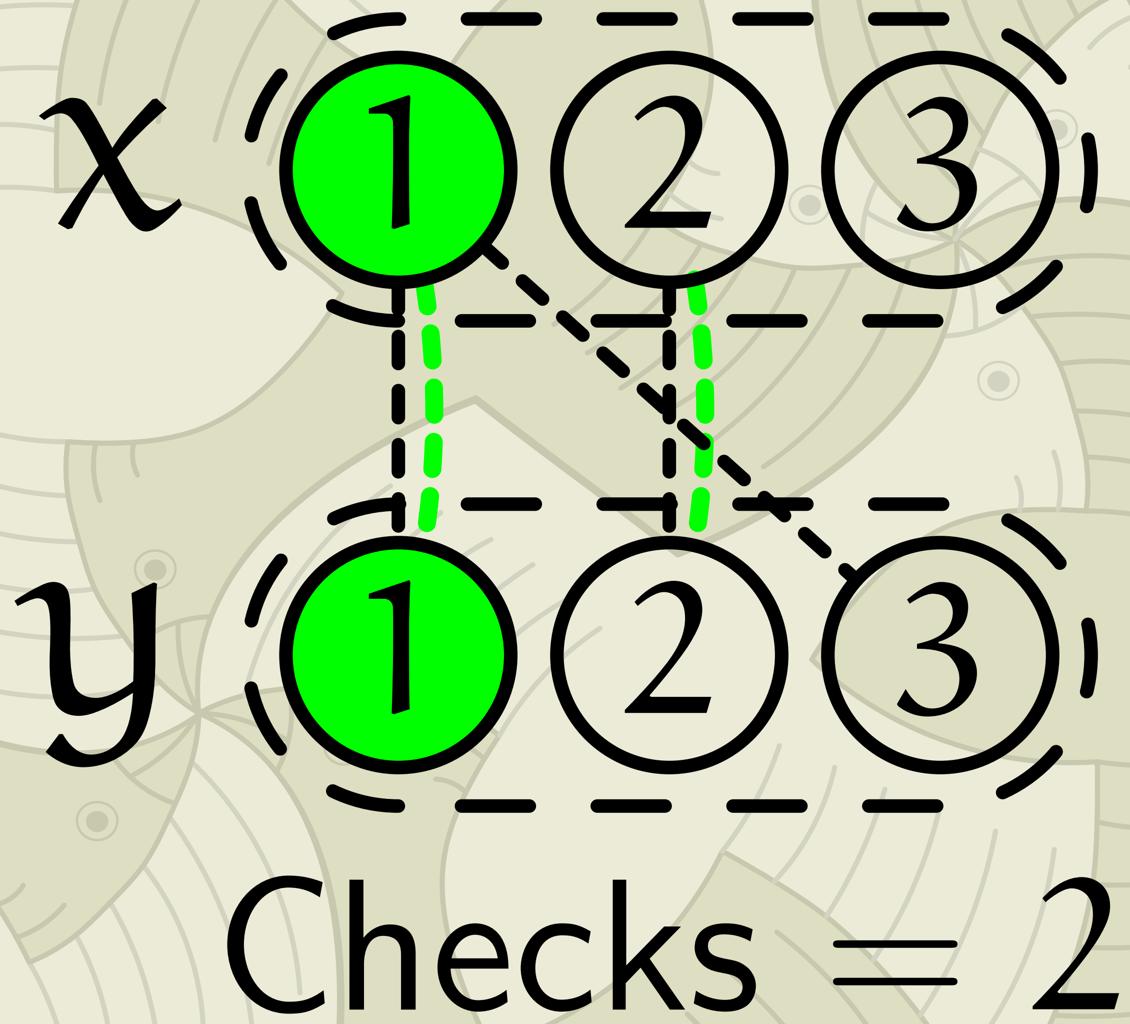


x
1
2
3
y
1
2
3
Checks = 0

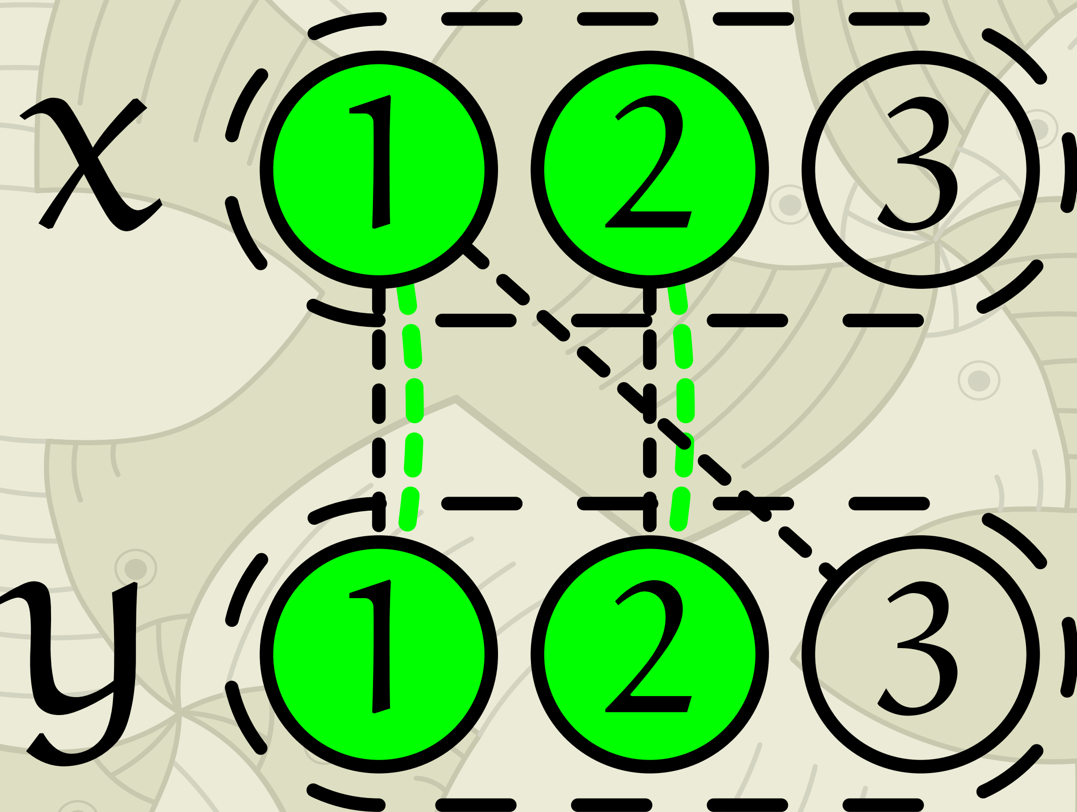


Checks = 1

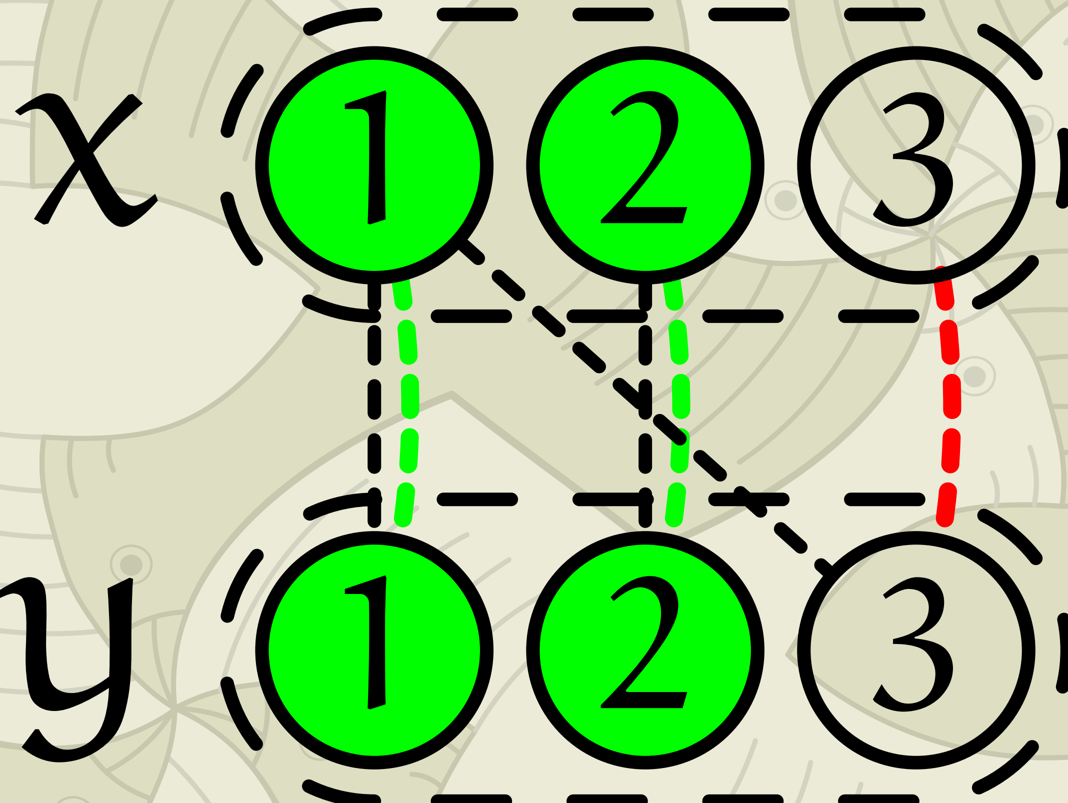

x
1
2
3
y
1
2
3
Checks = 1



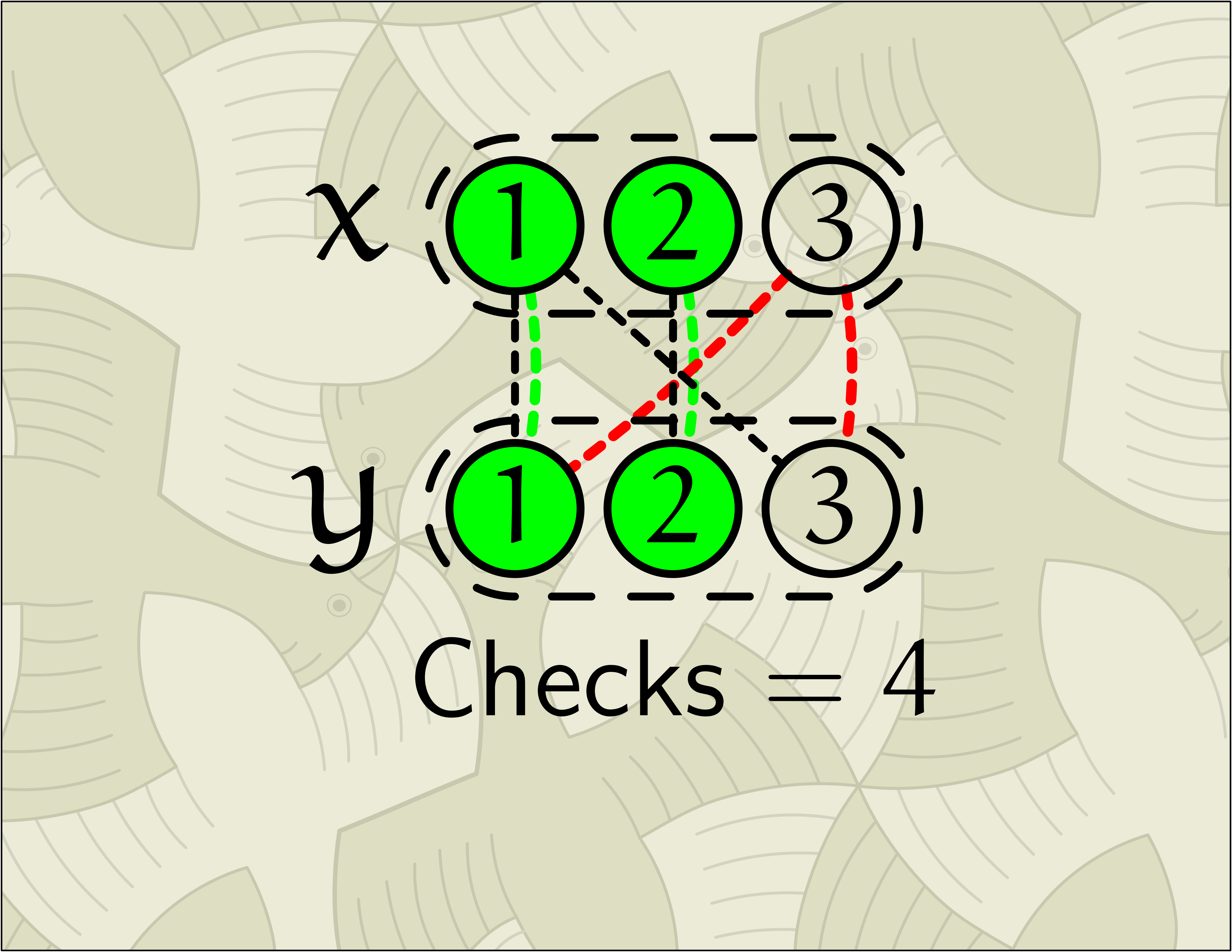
x
1
2
3
y
1
2
3
Checks = 2



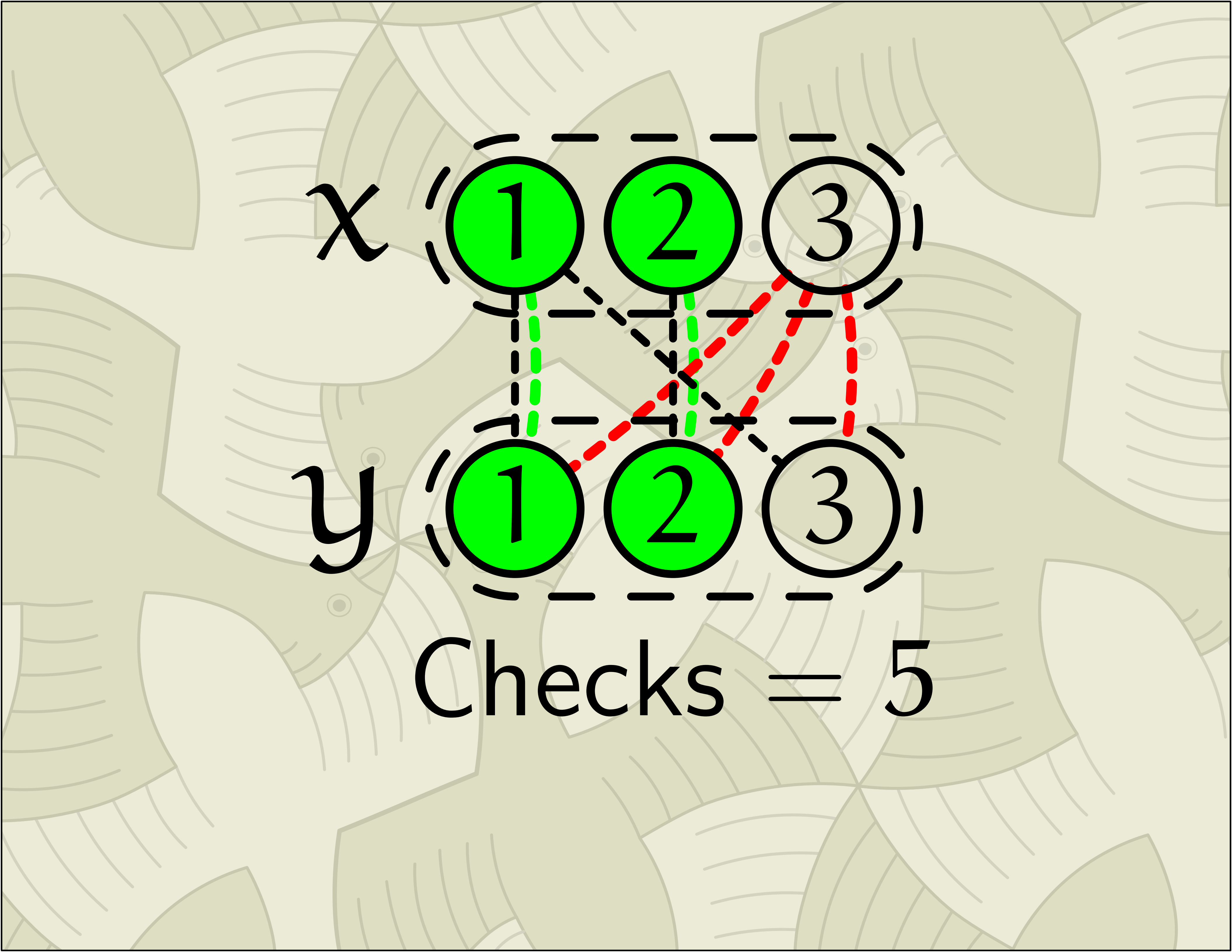
Checks $= 2$



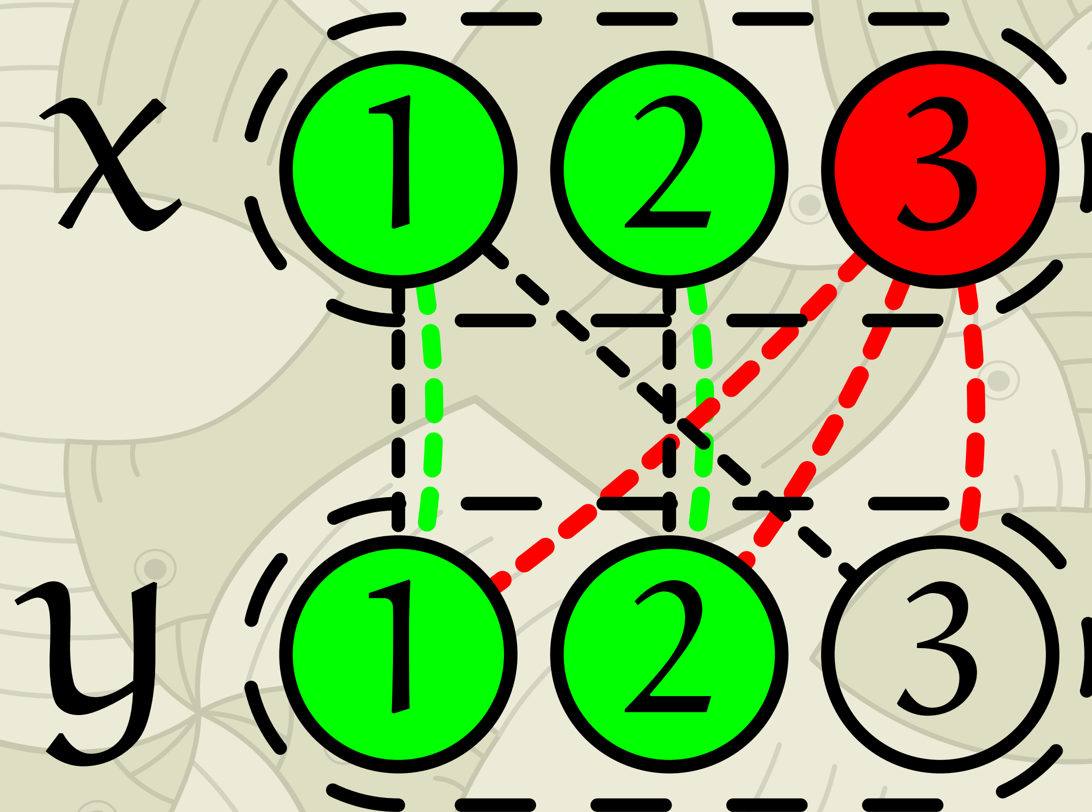
Checks $= 3$



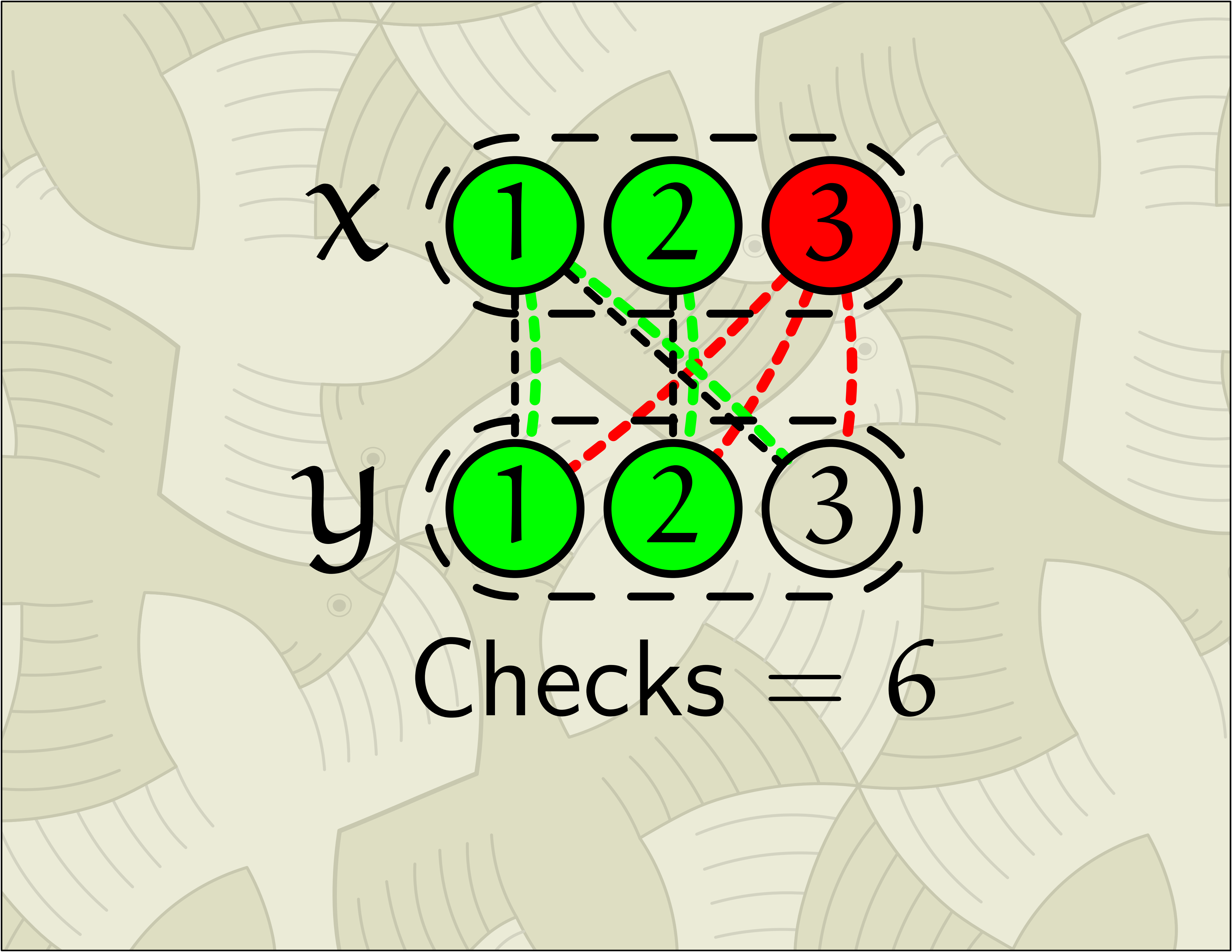
x
1
2
3
y
1
2
3
Checks = 4

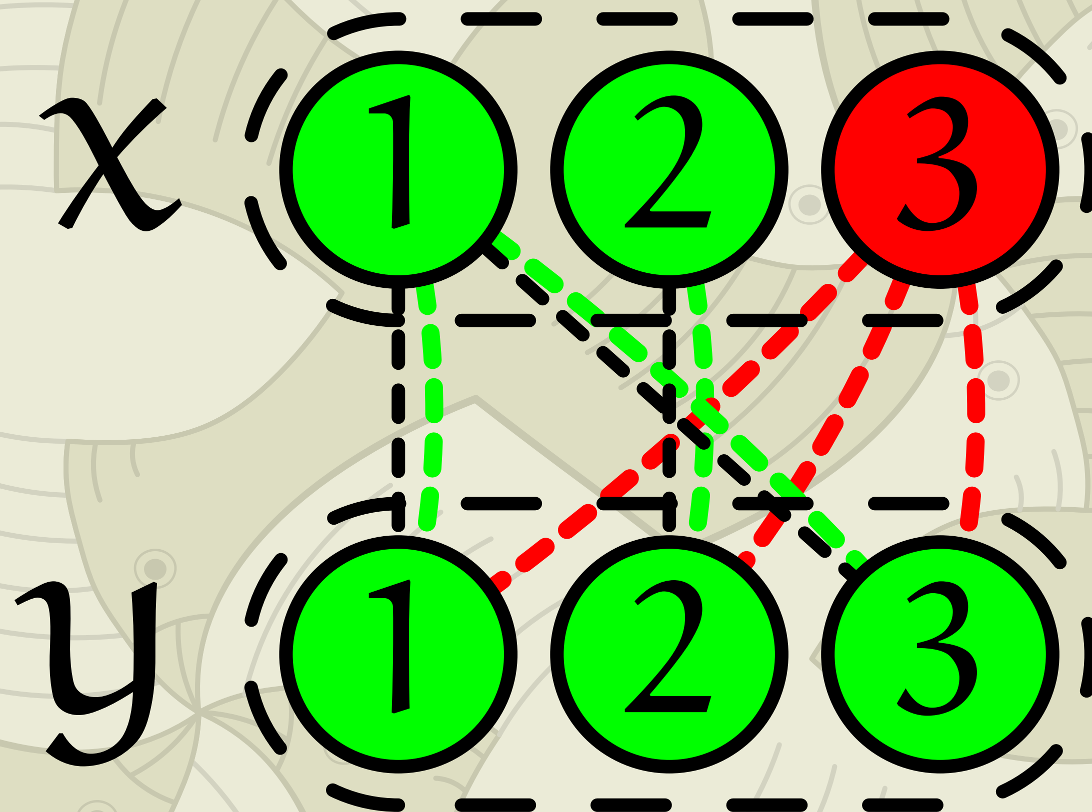


x
1
2
3
y
1
2
3
Checks = 5



Checks $= 5$


x
1
2
3
y
1
2
3
Checks = 6



Checks = 6

Case Study

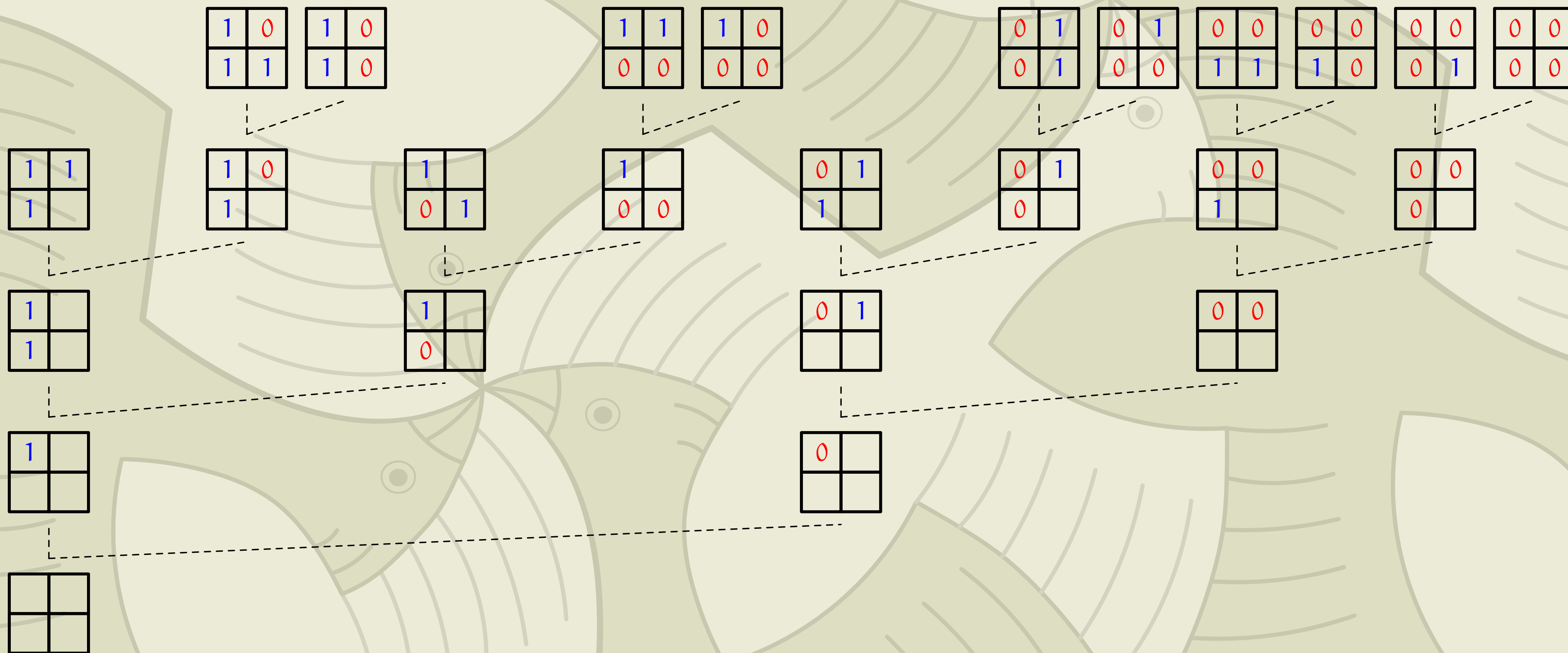
Let's study $\mathcal{L}$ and $\mathcal{D}$ if there are only two values in the domains of the variables.

We shall assume there are two variables x and y. The size of the domain of x will be a and that of y will be b.

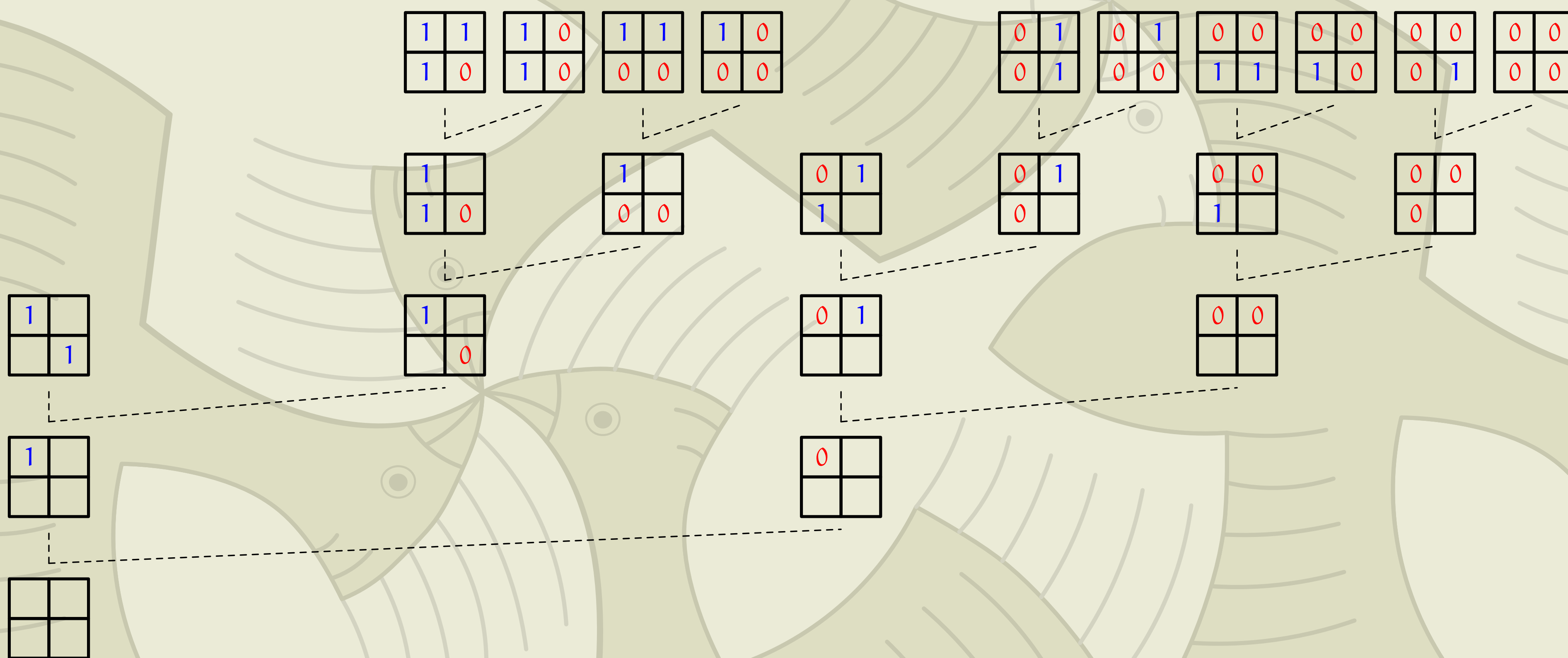
A constraint M between x and y is a by b zero-one matrix. M allows the "simultaneous assignment" $x = i$ and $y = j$ if and only if $M_{ij} = 1$.

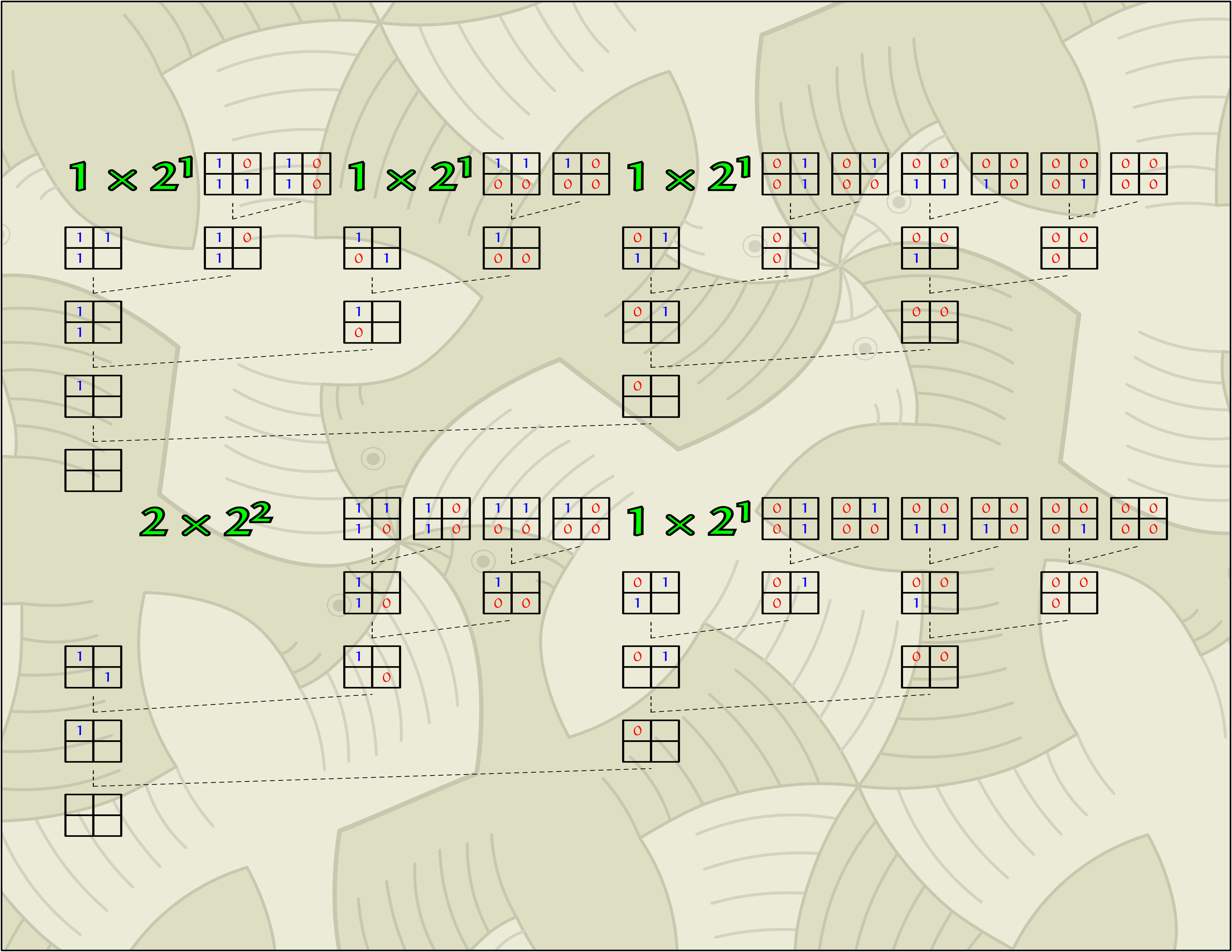
Our objective is to find out, for each column i and each row j, if there's a 1 in the i-row and the j-th column.

Traces of $\mathcal{L}$ for the Two by Two Case



Traces of $\mathcal{D}$ for the Two by Two Case




1 × 2¹
1 × 2¹
1 × 2¹
2 × 2²
1 × 2¹

Results

Definition 1. *Let* a *and* b *be positive integers. The set containing all* a *by* b *constraints will be denoted by* $\mathbb{M}^{ab}$*.*

Definition 2. *Let* $\mathcal{A}$ *be an arc-consistency algorithm and let* M *be a constraint between* x *and* y*. The number of checks required by* $\mathcal{A}$ *to remove the unsupported values from the domains of* x *and* y *will be denoted* $\mathrm{checks}_{\mathcal{A}}(M)$*.*

Definition 3. [Average Time-Complexity] *Let* $\mathcal{A}$ *be an arc-consistency algorithm. The* average time-complexity *of* $\mathcal{A}$ over $\mathbb{M}^{ab}$ *is the function* $\text{avg}_{\mathcal{A}} : \mathbb{N} \times \mathbb{N} \mapsto \mathbb{Q}$, *where*

$$\text{avg}_{\mathcal{A}}(a, b) = \sum_{M \in \mathbb{M}^{ab}} \text{checks}_{\mathcal{A}}(M)/2^{ab}.$$

Average Time-Complexity Results for $\mathcal{L}$

Theorem 1. [Average Time Complexity of $\mathcal{L}$] *The average time complexity of $\mathcal{L}$ over $\mathbb{M}^{ab}$ is given by:*

$$\mathrm{avg}_{\mathcal{L}}(a,b) = a(2-2^{1-b}) + (1-b)2^{1-a} + 2\sum_{c=2}^{b}(1-2^{-c})^{a}.$$

Following Flajolet and Sedgewick we obtain the following estimate:

$$\mathrm{avg}_{\mathcal{L}}(a,b) \approx \widetilde{\mathrm{avg}}_{\mathcal{L}}(a,b) = 2a + 2b - 2\log_2(a) - 0.665492.$$

For $a = b = 10$ we have

$$|\mathrm{avg}_{\mathcal{L}}(a,b) - \widetilde{\mathrm{avg}}_{\mathcal{L}}(a,b)| / \mathrm{avg}_{\mathcal{L}}(a,b) < 0.5\%.$$

Intuitive Proof for $\mathcal{L}$'s Bound

$\mathcal{L}$ requires about $2(a+b-1)$ checks to find support for the members of $D(x)$ and $D(y)$.

How to Find ι

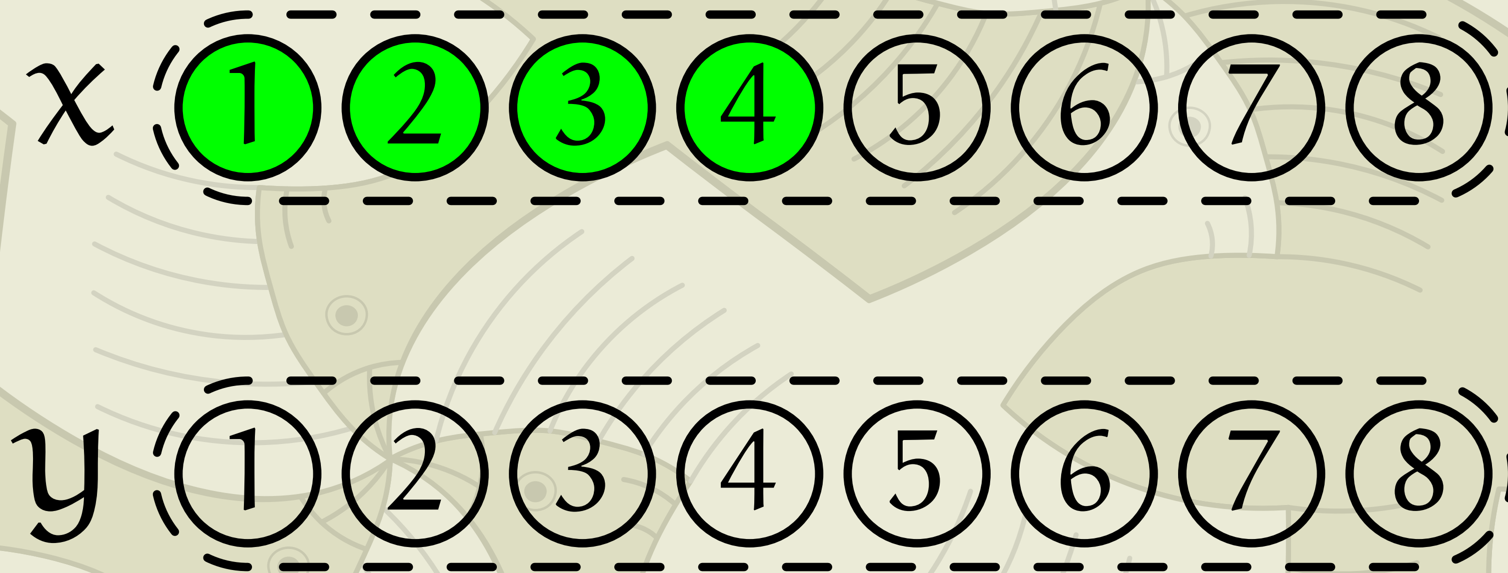
x ① ② ③ ④ ⑤ ⑥ ⑦ ⑧

y ① ② ③ ④ ⑤ ⑥ ⑦ ⑧

How to Find ι

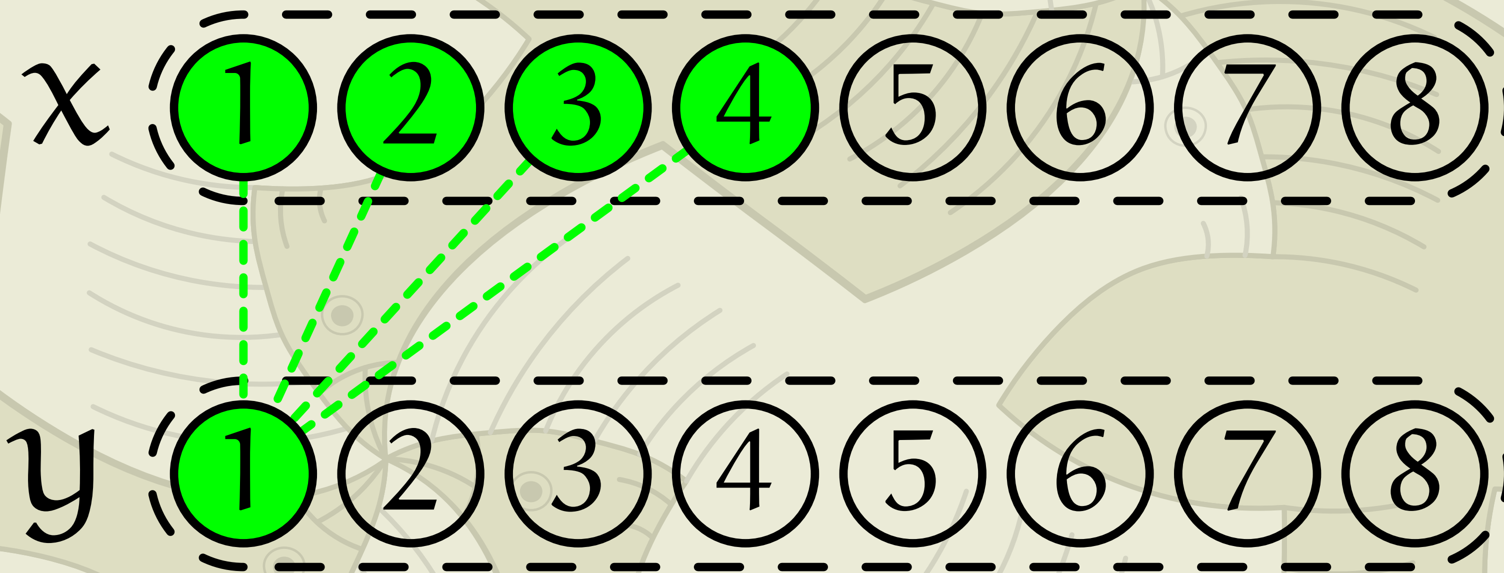
Let's assume that $|D(x)| = 2^k$, for some integer k.

How to Find ι



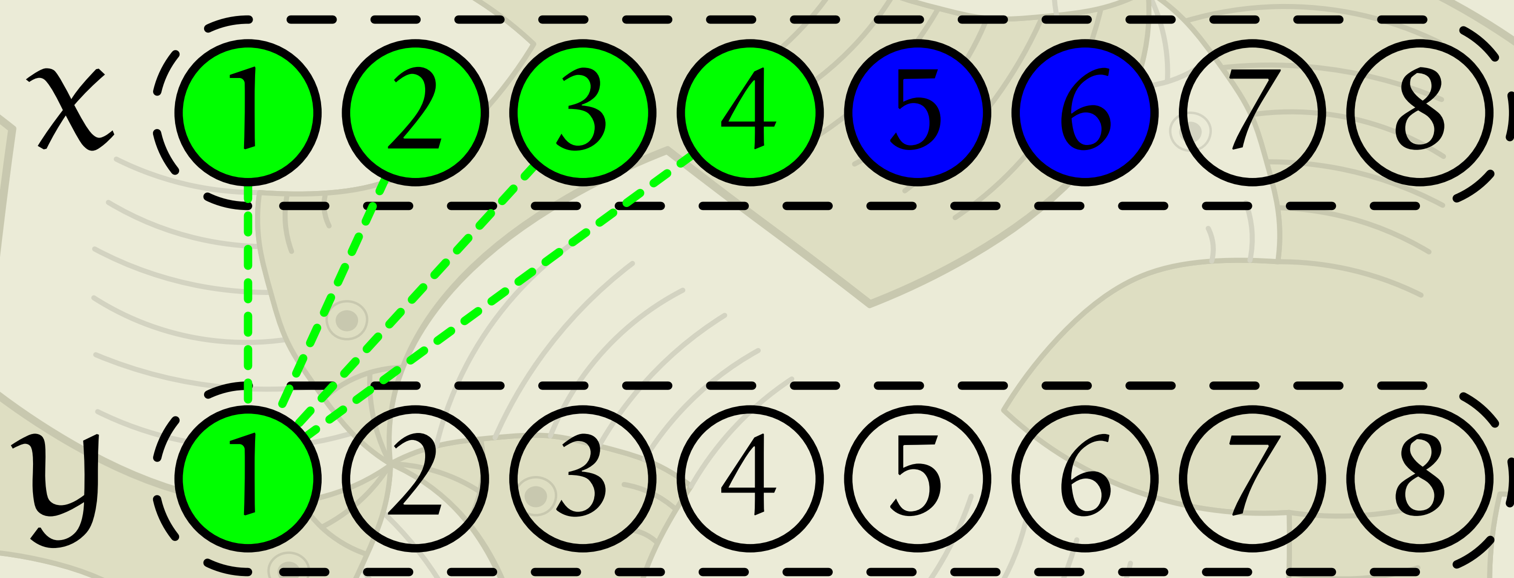
For about half of the members of $D(x)$

How to Find ι



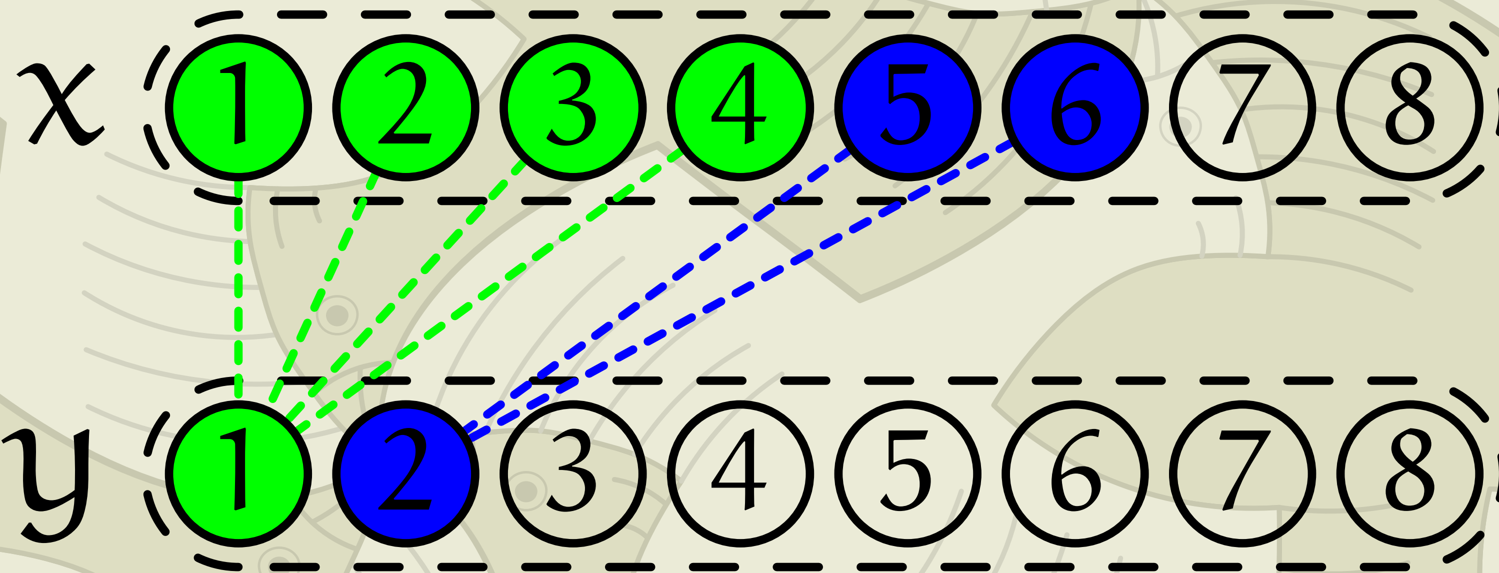
For about half of the members of $D(x)$ the first check succeeds.

How to Find ι



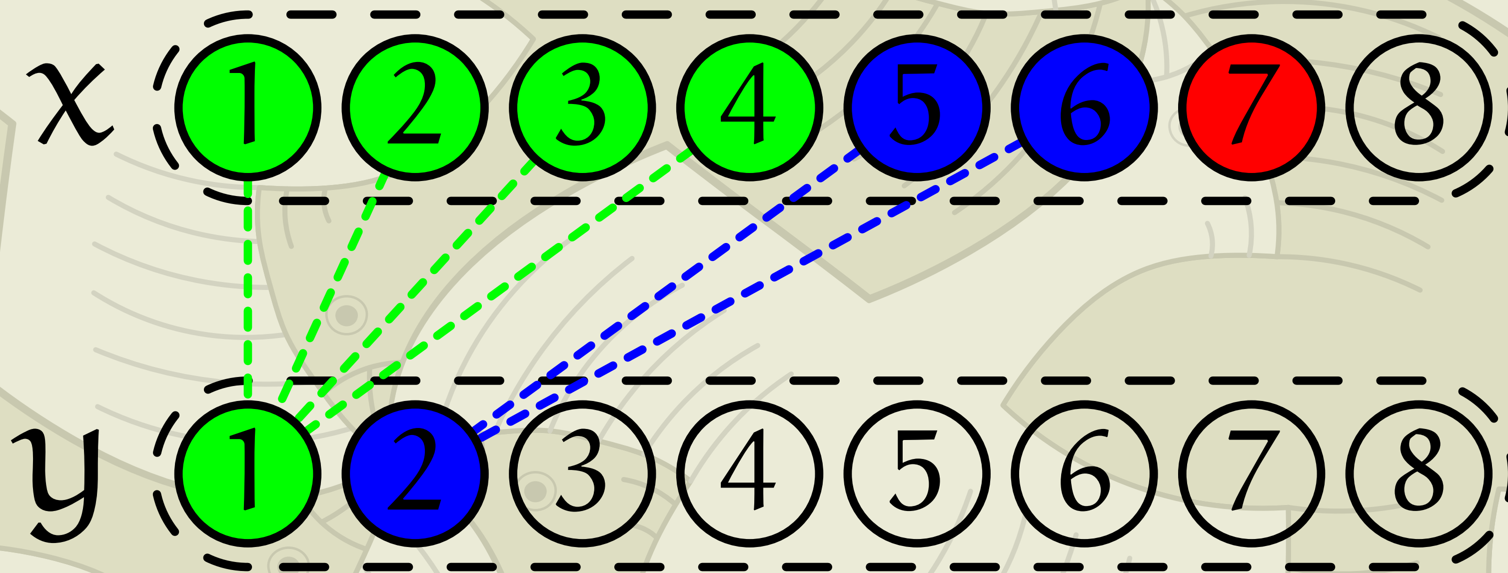
For about half of the remaining half

How to Find ι



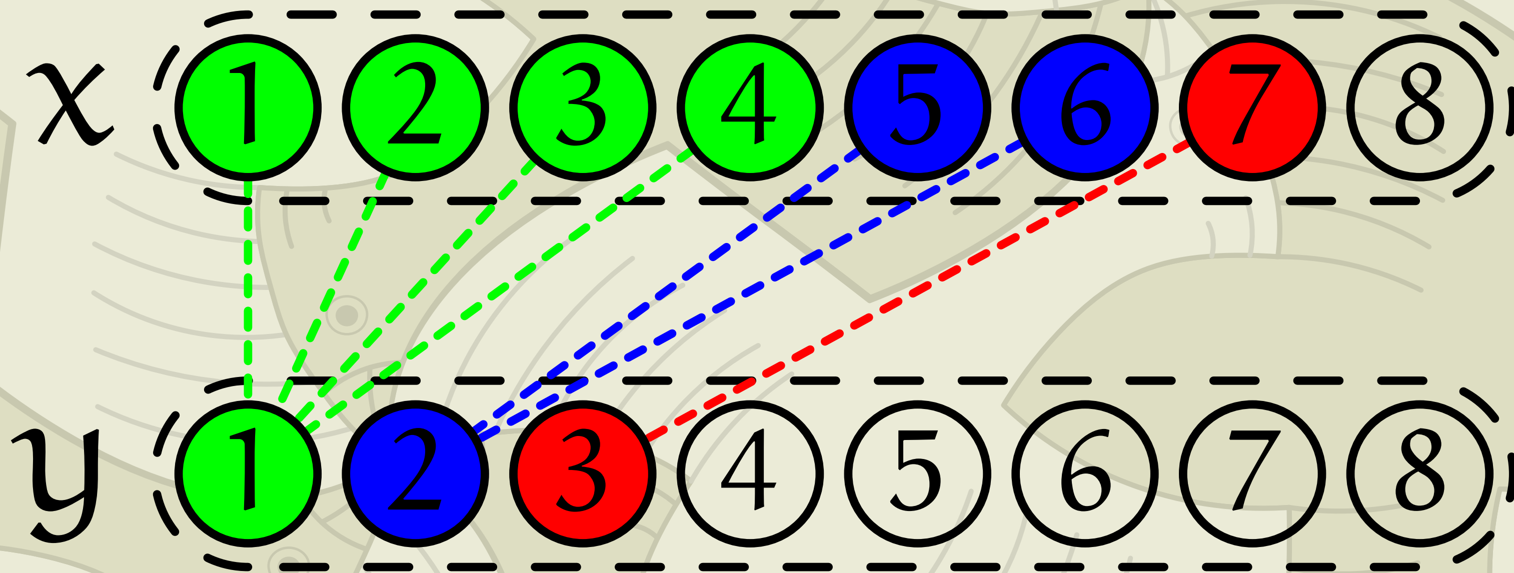
For about half of the remaining half the second check succeeds.

How to Find ι



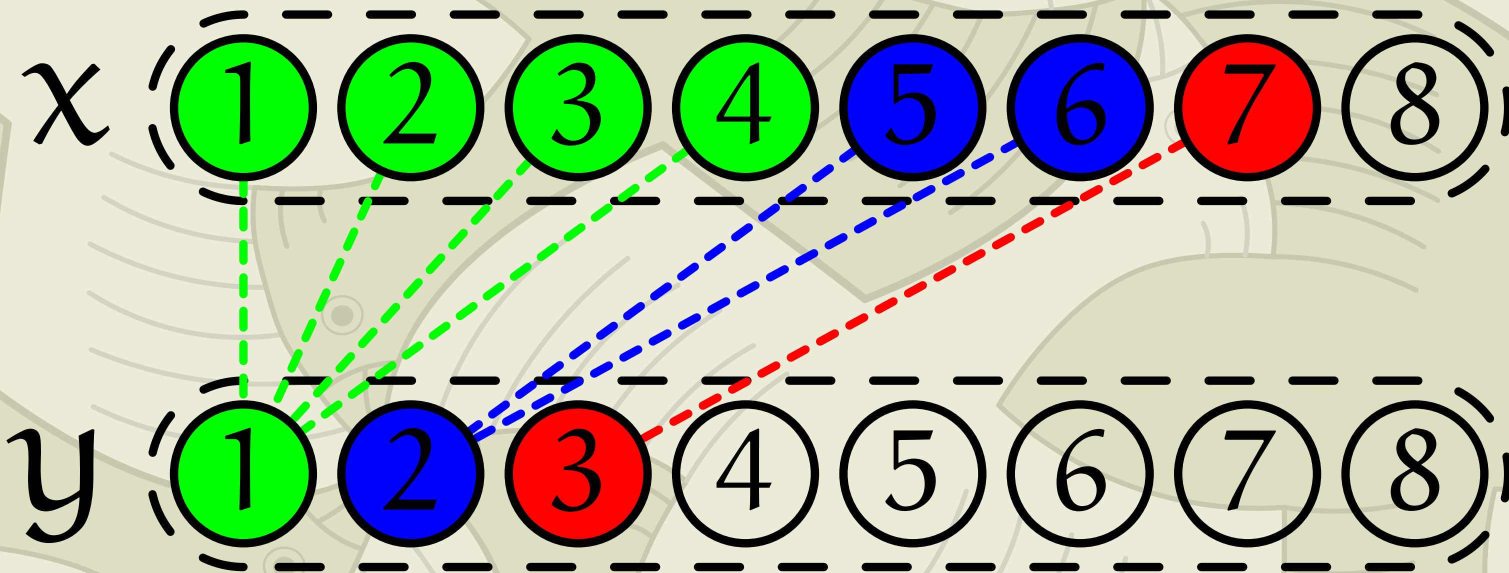
. . . . For one of the remaining two

How to Find l



. . . . For one of the remaining two the "l-th" check succeeds.

How to Find l



$|D(x)| \approx 1 + 2^0 + 2^1 + \cdots + 2^{l-1} = 2^l$. Therefore, $l \approx \log_2(|D(x)|)$.

Conjecture

If $0 < p/q < 1$ checks succeed on average then $\mathcal{L}$ will require about

$$q/p(a + b - \log_{q/p}(a))$$

checks on average.

Average Time-Complexity Results for $\mathcal{D}$

Theorem 2. [Average Time Complexity of $\mathcal{D}$] *The average time complexity of $\mathcal{D}$ over $\mathbb{M}^{ab}$ is* exactly $\operatorname{avg}_{\mathcal{D}}(a, b)$, *where* $\operatorname{avg}_{\mathcal{D}}(a, 0) = \operatorname{avg}_{\mathcal{D}}(0, b) = 0$, *and*

$$\begin{aligned} \operatorname{avg}_{\mathcal{D}}(a, b) \; = \; & 2 + (b-2)2^{1-a} + (a-2)2^{1-b} + 2^{2-a-b} \\ & - (a-1)2^{1-2b} + 2^{-b} \operatorname{avg}_{\mathcal{D}}(a-1, b) \\ & + (1-2^{-b}) \operatorname{avg}_{\mathcal{D}}(a-1, b-1) \end{aligned}$$

if $a \neq 0$ *and* $b \neq 0$.

From this we can derive the following bound:

$$\begin{aligned}\mathrm{avg}_{\mathcal{D}}(\mathfrak{a},\mathfrak{b}) \quad < \quad & 2\max(\mathfrak{a},\mathfrak{b})+2 \\ & -(2\max(\mathfrak{a},\mathfrak{b})+\min(\mathfrak{a},\mathfrak{b}))2^{-\min(\mathfrak{a},\mathfrak{b})} \\ & -(2\min(\mathfrak{a},\mathfrak{b})+3\max(\mathfrak{a},\mathfrak{b}))2^{-\max(\mathfrak{a},\mathfrak{b})}.\end{aligned}$$

This bound is almost as good as you can get.

Discussion

- Up till recently it was a common belief that domain-heuristics have little—if not no—effect on the performance of arc-consistency algorithms. This belief is simply *not* true;
- Proof has been presented that $\mathcal{D}$ is better than $\mathcal{L}$;
- Evidence has been presented that $\mathcal{D}$ is "good;"
- Arc-consistency algorithms should prefer double-support checks at domain level.

Future Work

1. Incorporate the double-support heuristic into an algorithm which does not repeat support-checks;

2. Study the case where the average tightness differs from $1/2$;

3. Study the effects that arc-heuristics have on the *average* time-complexity of arc-consistency algorithms;

4. Generalise the notion of double-support check for arc-consistency to higher-order consistency.

Questions Anybody?