Constraint Programming
– Filtering : Part 1 –

Christophe Lecoutre
lecoutre@cril.fr

CRIL-CNRS UMR 8188
Universite d'Artois
Lens, France

January 2021

Outline

1. Filtering Domains with Constraints

2. Principle of Constraint Propagation

Outline

1. Filtering Domains with Constraints

2. Principle of Constraint Propagation

Filtering Domains by means of Constraints

Each constraint represents a "sub-problem" from which some *inconsistent* values can be deleted.

Inconsistent values belong to no solution (of the sub-problem).

Several levels/types of filtering can be defined. For the moment, we only cite:

- AC (Arc Consistency) : all inconsistent values are identified and deleted
- BC (Bounds Consistency) : inconsistent values corresponding to the bounds of the domains are identified and deleted

Warning.

For non-binary constraints, AC is often denoted by GAC (but not in this course).

Filtering Domains by means of Constraints

Each constraint represents a "sub-problem" from which some *inconsistent* values can be deleted.

Inconsistent values belong to no solution (of the sub-problem).

Several levels/types of filtering can be defined. For the moment, we only cite:

- AC (Arc Consistency) : all inconsistent values are identified and deleted
- BC (Bounds Consistency) : inconsistent values corresponding to the bounds of the domains are identified and deleted

Warning.

For non-binary constraints, AC is often denoted by GAC (but not in this course).

Filtering Domains by means of Constraints

Each constraint represents a "sub-problem" from which some *inconsistent* values can be deleted.

Inconsistent values belong to no solution (of the sub-problem).

Several levels/types of filtering can be defined. For the moment, we only cite:

- AC (Arc Consistency) : all inconsistent values are identified and deleted
- BC (Bounds Consistency) : inconsistent values corresponding to the bounds of the domains are identified and deleted

Warning.

For non-binary constraints, AC is often denoted by GAC (but not in this course).

Filtering Domains by means of Constraints

Each constraint represents a "sub-problem" from which some *inconsistent* values can be deleted.

Inconsistent values belong to no solution (of the sub-problem).

Several levels/types of filtering can be defined. For the moment, we only cite:

- AC (Arc Consistency) : all inconsistent values are identified and deleted
- BC (Bounds Consistency) : inconsistent values corresponding to the bounds of the domains are identified and deleted

Warning.

For non-binary constraints, AC is often denoted by GAC (but not in this course).

Filtering Domains by means of Constraints

Each constraint represents a "sub-problem" from which some *inconsistent* values can be deleted.

Inconsistent values belong to no solution (of the sub-problem).

Several levels/types of filtering can be defined. For the moment, we only cite:

- AC (Arc Consistency) : all inconsistent values are identified and deleted
- BC (Bounds Consistency) : inconsistent values corresponding to the bounds of the domains are identified and deleted

Warning.

For non-binary constraints, AC is often denoted by GAC (but not in this course).

Filtering Domains by means of Constraints

Each constraint represents a "sub-problem" from which some *inconsistent* values can be deleted.

Inconsistent values belong to no solution (of the sub-problem).

Several levels/types of filtering can be defined. For the moment, we only cite:

- AC (Arc Consistency) : all inconsistent values are identified and deleted
- BC (Bounds Consistency) : inconsistent values corresponding to the bounds of the domains are identified and deleted

Warning.

For non-binary constraints, AC is often denoted by GAC (but not in this course).

Example.

Constraint $x < y$ with

- $dom(x) = 10..20$
- $dom(y) = 0..15$

After AC filtering, we obtain:

- $dom(x) = 10..14$
- $dom(y) = 11..15$

After BC filtering, we obtain:

- $dom(x) = 10..14$
- $dom(y) = 11..15$

Constraint $w + 3 = z$ with

- $dom(w) = \{1, 3, 4, 5\}$
- $dom(z) = \{4, 5, 8\}$

After AC filtering, we obtain:

- $dom(w) = \{1, 5\}$
- $dom(z) = \{4, 8\}$

After BC filtering, we obtain:

- $dom(w) = \{1, 3, 4, 5\}$
- $dom(z) = \{4, 5, 8\}$

Example.

Constraint $x < y$ with

- $dom(x) = 10..20$
- $dom(y) = 0..15$

After AC filtering, we obtain:

- $dom(x) = 10..14$
- $dom(y) = 11..15$

After BC filtering, we obtain:

- $dom(x) = 10..14$
- $dom(y) = 11..15$

Constraint $w + 3 = z$ with

- $dom(w) = \{1, 3, 4, 5\}$
- $dom(z) = \{4, 5, 8\}$

After AC filtering, we obtain:

- $dom(w) = \{1, 5\}$
- $dom(z) = \{4, 8\}$

After BC filtering, we obtain:

- $dom(w) = \{1, 3, 4, 5\}$
- $dom(z) = \{4, 5, 8\}$

Example.

Constraint $x < y$ with

- $dom(x) = 10..20$
- $dom(y) = 0..15$

After AC filtering, we obtain:

- $dom(x) = 10..14$
- $dom(y) = 11..15$

After BC filtering, we obtain:

- $dom(x) = 10..14$
- $dom(y) = 11..15$

Constraint $w + 3 = z$ with

- $dom(w) = \{1, 3, 4, 5\}$
- $dom(z) = \{4, 5, 8\}$

After AC filtering, we obtain:

- $dom(w) = \{1, 5\}$
- $dom(z) = \{4, 8\}$

After BC filtering, we obtain:

- $dom(w) = \{1, 3, 4, 5\}$
- $dom(z) = \{4, 5, 8\}$

Example.

Constraint $x < y$ with

- $dom(x) = 10..20$
- $dom(y) = 0..15$

After AC filtering, we obtain:

- $dom(x) = 10..14$
- $dom(y) = 11..15$

After BC filtering, we obtain:

- $dom(x) = 10..14$
- $dom(y) = 11..15$

Constraint $w + 3 = z$ with

- $dom(w) = \{1, 3, 4, 5\}$
- $dom(z) = \{4, 5, 8\}$

After AC filtering, we obtain:

- $dom(w) = \{1, 5\}$
- $dom(z) = \{4, 8\}$

After BC filtering, we obtain:

- $dom(w) = \{1, 3, 4, 5\}$
- $dom(z) = \{4, 5, 8\}$

Example.

Constraint $x < y$ with

- $dom(x) = 10..20$
- $dom(y) = 0..15$

After AC filtering, we obtain:

- $dom(x) = 10..14$
- $dom(y) = 11..15$

After BC filtering, we obtain:

- $dom(x) = 10..14$
- $dom(y) = 11..15$

Constraint $w + 3 = z$ with

- $dom(w) = \{1, 3, 4, 5\}$
- $dom(z) = \{4, 5, 8\}$

After AC filtering, we obtain:

- $dom(w) = \{1, 5\}$
- $dom(z) = \{4, 8\}$

After BC filtering, we obtain:

- $dom(w) = \{1, 3, 4, 5\}$
- $dom(z) = \{4, 5, 8\}$

Example.

Constraint $x < y$ with

- $dom(x) = 10..20$
- $dom(y) = 0..15$

After AC filtering, we obtain:

- $dom(x) = 10..14$
- $dom(y) = 11..15$

After BC filtering, we obtain:

- $dom(x) = 10..14$
- $dom(y) = 11..15$

Constraint $w + 3 = z$ with

- $dom(w) = \{1, 3, 4, 5\}$
- $dom(z) = \{4, 5, 8\}$

After AC filtering, we obtain:

- $dom(w) = \{1, 5\}$
- $dom(z) = \{4, 8\}$

After BC filtering, we obtain:

- $dom(w) = \{1, 3, 4, 5\}$
- $dom(z) = \{4, 5, 8\}$

Example.

Constraint $x < y$ with

- $dom(x) = 10..20$
- $dom(y) = 0..15$

After AC filtering, we obtain:

- $dom(x) = 10..14$
- $dom(y) = 11..15$

After BC filtering, we obtain:

- $dom(x) = 10..14$
- $dom(y) = 11..15$

Constraint $w + 3 = z$ with

- $dom(w) = \{1, 3, 4, 5\}$
- $dom(z) = \{4, 5, 8\}$

After AC filtering, we obtain:

- $dom(w) = \{1, 5\}$
- $dom(z) = \{4, 8\}$

After BC filtering, we obtain:

- $dom(w) = \{1, 3, 4, 5\}$
- $dom(z) = \{4, 5, 8\}$

Example.

Constraint $x < y$ with

- $dom(x) = 10..20$
- $dom(y) = 0..15$

After AC filtering, we obtain:

- $dom(x) = 10..14$
- $dom(y) = 11..15$

After BC filtering, we obtain:

- $dom(x) = 10..14$
- $dom(y) = 11..15$

Constraint $w + 3 = z$ with

- $dom(w) = \{1, 3, 4, 5\}$
- $dom(z) = \{4, 5, 8\}$

After AC filtering, we obtain:

- $dom(w) = \{1, 5\}$
- $dom(z) = \{4, 8\}$

After BC filtering, we obtain:

- $dom(w) = \{1, 3, 4, 5\}$
- $dom(z) = \{4, 5, 8\}$

Example.

Constraint $x < y$ with

- $dom(x) = 10..20$
- $dom(y) = 0..15$

After AC filtering, we obtain:

- $dom(x) = 10..14$
- $dom(y) = 11..15$

After BC filtering, we obtain:

- $dom(x) = 10..14$
- $dom(y) = 11..15$

Constraint $w + 3 = z$ with

- $dom(w) = \{1, 3, 4, 5\}$
- $dom(z) = \{4, 5, 8\}$

After AC filtering, we obtain:

- $dom(w) = \{1, 5\}$
- $dom(z) = \{4, 8\}$

After BC filtering, we obtain:

- $dom(w) = \{1, 3, 4, 5\}$
- $dom(z) = \{4, 5, 8\}$

Example.

Constraint $x < y$ with

- $dom(x) = 10..20$
- $dom(y) = 0..15$

After AC filtering, we obtain:

- $dom(x) = 10..14$
- $dom(y) = 11..15$

After BC filtering, we obtain:

- $dom(x) = 10..14$
- $dom(y) = 11..15$

Constraint $w + 3 = z$ with

- $dom(w) = \{1, 3, 4, 5\}$
- $dom(z) = \{4, 5, 8\}$

After AC filtering, we obtain:

- $dom(w) = \{1, 5\}$
- $dom(z) = \{4, 8\}$

After BC filtering, we obtain:

- $dom(w) = \{1, 3, 4, 5\}$
- $dom(z) = \{4, 5, 8\}$

Notion of Support

For a constraint c

- an allowed tuple, or tuple accepted by c, is an element of $T = rel(c)$
- a valid tuple is an element of $V = \Pi_{x \in scp(c)} dom(x)$
- a support (on c) is a tuple that is both allowed and valid, i.e., an element of $T \cap V$

Remark.
A support on c is what we have previously informally called a solution of the "sub-problem" c.

Notion of Support

For a constraint c

- an allowed tuple, or tuple accepted by c, is an element of $T = rel(c)$
- a valid tuple is an element of $V = \Pi_{x \in scp(c)} dom(x)$
- a support (on c) is a tuple that is both allowed and valid, i.e., an element of $T \cap V$

Remark.

A support on c is what we have previously informally called a solution of the "sub-problem" c.

Notion of Support

For a constraint c

- an allowed tuple, or tuple accepted by c, is an element of $T = rel(c)$
- a valid tuple is an element of $V = \Pi_{x \in scp(c)} dom(x)$
- a support (on c) is a tuple that is both allowed and valid, i.e., an element of $T \cap V$

Remark.

A support on c is what we have previously informally called a solution of the "sub-problem" c.

Notion of Support

For a constraint c

- an allowed tuple, or tuple accepted by c, is an element of $T = rel(c)$
- a valid tuple is an element of $V = \Pi_{x \in scp(c)} dom(x)$
- a support (on c) is a tuple that is both allowed and valid, i.e., an element of $T \cap V$

Remark.

A support on c is what we have previously informally called a solution of the "sub-problem" c.

Notion of Support

For a constraint c

- an allowed tuple, or tuple accepted by c, is an element of $T = rel(c)$
- a valid tuple is an element of $V = \Pi_{x \in scp(c)} dom(x)$
- a support (on c) is a tuple that is both allowed and valid, i.e., an element of $T \cap V$

Remark.

A support on c is what we have previously informally called a solution of the “sub-problem” c.

Notion of Support

Example.

Let c_{xyz} be a ternary constraint, and let us suppose that $dom(x) = dom(y) = \{a, b\}$ and $dom(z) = \{b, c\}$. We have:

- $T = rel(c_{xyz})$
- $V = dom(x) \times dom(y) \times dom(z)$

T
a a a
a b b
a c c
b a a
b b b
c a a
c c c

$\cap$

V
a a b
a a c
a b b
a b c
b a b
b a c
b b b
b b c

Notion of Support

Example.

Let c_{xyz} be a ternary constraint, and let us suppose that $dom(x) = dom(y) = \{a, b\}$ and $dom(z) = \{b, c\}$. We have:

- $T = rel(c_{xyz})$
- $V = dom(x) \times dom(y) \times dom(z)$

T
a a a
a b b
a c c
b a a
b b b
c a a
c c c

$\cap$

V
a a b
a a c
a b b
a b c
b a b
b a c
b b b
b b c

Is there a support for (z, b)?

Notion of Support

Example.

Let c_{xyz} be a ternary constraint, and let us suppose that $dom(x) = dom(y) = \{a, b\}$ and $dom(z) = \{b, c\}$. We have:

- $T = rel(c_{xyz})$
- $V = dom(x) \times dom(y) \times dom(z)$

T
a a a
a b b
a c c
b a a
b b b
c a a
c c c

$\cap$

V
a a b
a a c
a b b
a b c
b a b
b a c
b b b
b b c

(z, b) has a support ✓

Notion of Support

Example.

Let c_{xyz} be a ternary constraint, and let us suppose that $dom(x) = dom(y) = \{a, b\}$ and $dom(z) = \{b, c\}$. We have:

- $T = rel(c_{xyz})$
- $V = dom(x) \times dom(y) \times dom(z)$

T
a a a
a b b
a c c
b a a
b b b
c a a
c c c

$\cap$

V
a a b
a a c
a b b
a b c
b a b
b a c
b b b
b b c

Is there a support for (z, c)?

Example.

Let c_{xyz} be a ternary constraint, and let us suppose that $dom(x) = dom(y) = \{a, b\}$ and $dom(z) = \{b, c\}$. We have:

- $T = rel(c_{xyz})$
- $V = dom(x) \times dom(y) \times dom(z)$

T
a a a
a b b
a c c
b a a
b b b
c a a
c c c

$\cap$

V
a a b
a a c
a b b
a b c
b a b
b a c
b b b
b b c

(z, c) has no support ✗

Arc Consistency (AC)

Definition

A constraint c is arc-consistent (AC) iff $\forall x \in scp(c)$, $\forall a \in dom(x)$, there exists a support of (x, a) on c, i.e., a support τ on c such that $\tau[x] = a$.

Example.

Let x and y be two variables such that $dom(x) = dom(y) = \{1, 2\}$, and let $x = y$ be a binary constraint.

- the tuple $\tau = (1, 2)$ // $\tau[x] = 1 \wedge \tau[y] = 2$
  - is valid
  - but not accepted by $x = y$
- the tuple $\tau = (3, 3)$
  - is not valid,
  - but accepted by $x = y$
- the tuple $\tau = (2, 2)$
  - is valid
  - and accepted by $x = y$

  it represents a support of both $(x, 2)$ and $(y, 2)$ on $x = y$

Arc Consistency (AC)

Definition

A constraint c is arc-consistent (AC) iff $\forall x \in scp(c)$, $\forall a \in dom(x)$, there exists a support of (x, a) on c, i.e., a support τ on c such that $\tau[x] = a$.

Example.

Let x and y be two variables such that $dom(x) = dom(y) = \{1, 2\}$, and let $x = y$ be a binary constraint.

- the tuple $\tau = (1, 2)$ // $\tau[x] = 1 \wedge \tau[y] = 2$
  - is valid
  - but not accepted by $x = y$
- the tuple $\tau = (3, 3)$
  - is not valid,
  - but accepted by $x = y$
- the tuple $\tau = (2, 2)$
  - is valid
  - and accepted by $x = y$

  it represents a support of both $(x, 2)$ and $(y, 2)$ on $x = y$

Arc Consistency (AC)

Definition

A constraint c is arc-consistent (AC) iff $\forall x \in scp(c)$, $\forall a \in dom(x)$, there exists a support of (x, a) on c, i.e., a support τ on c such that $\tau[x] = a$.

Example.

Let x and y be two variables such that $dom(x) = dom(y) = \{1, 2\}$, and let $x = y$ be a binary constraint.

- the tuple $\tau = (1, 2)$ // $\tau[x] = 1 \wedge \tau[y] = 2$
  - is valid
  - but not accepted by $x = y$
- the tuple $\tau = (3, 3)$
  - is not valid,
  - but accepted by $x = y$
- the tuple $\tau = (2, 2)$
  - is valid
  - and accepted by $x = y$

  it represents a support of both $(x, 2)$ and $(y, 2)$ on $x = y$

Arc Consistency (AC)

Definition

A constraint c is arc-consistent (AC) iff $\forall x \in scp(c)$, $\forall a \in dom(x)$, there exists a support of (x, a) on c, i.e., a support τ on c such that $\tau[x] = a$.

Example.

Let x and y be two variables such that $dom(x) = dom(y) = \{1, 2\}$, and let $x = y$ be a binary constraint.

- the tuple $\tau = (1, 2)$ // $\tau[x] = 1 \wedge \tau[y] = 2$
  - is valid
  - but not accepted by $x = y$
- the tuple $\tau = (3, 3)$
  - is not valid,
  - but accepted by $x = y$
- the tuple $\tau = (2, 2)$
  - is valid
  - and accepted by $x = y$

  it represents a support of both $(x, 2)$ and $(y, 2)$ on $x = y$

Arc Consistency (AC)

Definition

A constraint c is arc-consistent (AC) iff $\forall x \in scp(c)$, $\forall a \in dom(x)$, there exists a support of (x, a) on c, i.e., a support τ on c such that $\tau[x] = a$.

Example.

Let x and y be two variables such that $dom(x) = dom(y) = \{1, 2\}$, and let $x = y$ be a binary constraint.

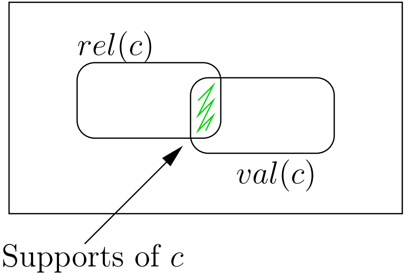
- the tuple $\tau = (1, 2)$ // $\tau[x] = 1 \wedge \tau[y] = 2$
  - is valid
  - but not accepted by $x = y$
- the tuple $\tau = (3, 3)$
  - is not valid,
  - but accepted by $x = y$
- the tuple $\tau = (2, 2)$
  - is valid
  - and accepted by $x = y$

  it represents a support of both $(x, 2)$ and $(y, 2)$ on $x = y$

Supports

In other words, the supports on a constraint c are those tuples that are present in the intersection of :

- the set of allowed tuples: $rel(c)$
- the set of valid tuples: $val(c) = \Pi_{x \in scp(c)} dom(x)$

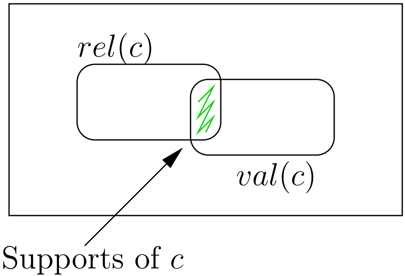


$\Rightarrow$ We need to "identify" these supports for filtering

Supports

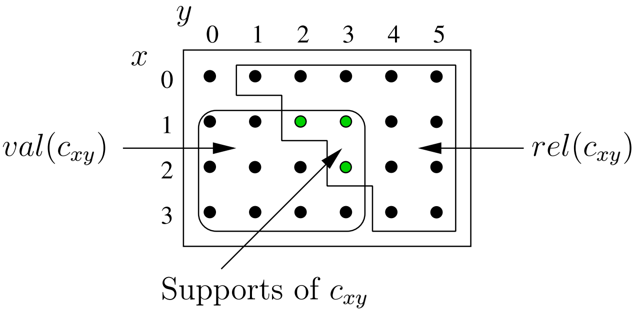
In other words, the supports on a constraint c are those tuples that are present in the intersection of :

- the set of allowed tuples: $rel(c)$
- the set of valid tuples: $val(c) = \Pi_{x \in scp(c)} dom(x)$



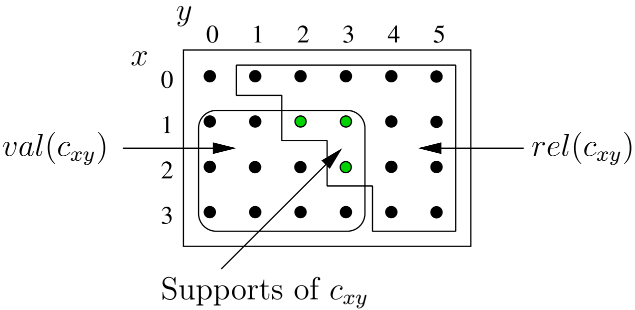
$\Rightarrow$ We need to "identify" these supports for filtering

Example



After AC filtering, we obtain?

Example



After AC filtering, we obtain?

AC Algorithm

Definition

A value (x, a) is *arc-inconsistent* on a constraint c when there is no support of (x, a) on c.

Definition

An AC algorithm for a constraint c is an algorithm that removes all values that are arc-inconsistent on c; the algorithm is said to enforce/establish AC on c.

Here is an AC algorithm that can be used in theory with any constraint c.

Algorithm 1: filterAC(c: Constraint)

for *each variable* $x \in scp(c)$ **do**
  for *each value* $a \in dom(x)$ **do**
    if $\neg$ *seekSupport*(c, x, a) `// function to be implemented` **then**
      remove a from $dom(x)$

AC Algorithm

Definition

A value (x, a) is *arc-inconsistent* on a constraint c when there is no support of (x, a) on c.

Definition

An AC algorithm for a constraint c is an algorithm that removes all values that are arc-inconsistent on c; the algorithm is said to enforce/establish AC on c.

Here is an AC algorithm that can be used in theory with any constraint c.

Algorithm 2: filterAC(c: Constraint)

```
for each variable x ∈ scp(c) do
    for each value a ∈ dom(x) do
        if ¬ seekSupport(c, x, a)    // function to be implemented
        then
            remove a from dom(x)
```

AC Algorithm

Definition

A value (x, a) is *arc-inconsistent* on a constraint c when there is no support of (x, a) on c.

Definition

An AC algorithm for a constraint c is an algorithm that removes all values that are arc-inconsistent on c; the algorithm is said to enforce/establish AC on c.

Here is an AC algorithm that can be used in theory with any constraint c.

Algorithm 3: filterAC(c: Constraint)

for *each variable* $x \in scp(c)$ **do**
    for *each value* $a \in dom(x)$ **do**
        if $\neg$ *seekSupport*(c, x, a) `// function to be implemented`
        then
            remove a from $dom(x)$

AC Filtering for `allDifferent`

Proposition

A constraint `allDifferent`*(X) is AC iff* $\forall X' \subseteq X$,

$$|dom(X')| = |X'| \Rightarrow \forall x \in X \setminus X', dom(x) = dom(x) \setminus dom(X')$$

where $dom(X') = \cup_{x' \in X'} dom(x')$

Remark.

A subset X' of variables such that $|dom(X')| = |X'|$ is called a Hall set.

Example.

The set of variables $\{x, y, z\}$ such that:

- $dom(x) = \{a, b\}$,
- $dom(y) = \{a, c\}$
- and $dom(z) = \{b, c\}$

is a Hall set (of size 3).

AC Filtering for `allDifferent`

Proposition

A constraint `allDifferent`(X) *is AC iff* $\forall X' \subseteq X$,

$$|dom(X')| = |X'| \Rightarrow \forall x \in X \setminus X', dom(x) = dom(x) \setminus dom(X')$$

where $dom(X') = \cup_{x' \in X'} dom(x')$

Remark.

A subset X' of variables such that $|dom(X')| = |X'|$ is called a Hall set.

Example.

The set of variables $\{x, y, z\}$ such that:

- $dom(x) = \{a, b\}$,
- $dom(y) = \{a, c\}$
- and $dom(z) = \{b, c\}$

is a Hall set (of size 3).

AC Filtering for `allDifferent`

Proposition

A constraint `allDifferent`(X) *is AC iff* $\forall X' \subseteq X$,

$$|dom(X')| = |X'| \Rightarrow \forall x \in X \setminus X', dom(x) = dom(x) \setminus dom(X')$$

where $dom(X') = \cup_{x' \in X'} dom(x')$

Remark.

A subset X' of variables such that $|dom(X')| = |X'|$ is called a Hall set.

Example.

The set of variables $\{x, y, z\}$ such that:

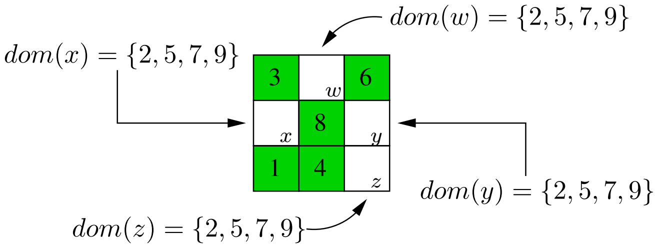
- $dom(x) = \{a, b\}$,
- $dom(y) = \{a, c\}$
- and $dom(z) = \{b, c\}$

is a Hall set (of size 3).

AC Filtering for `allDifferent`

Example.

For a Sudoku block, a constraint `allDifferent`(w, x, y, z) :

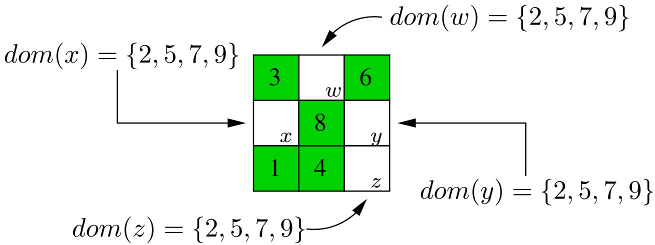


Can we filter?

AC Filtering for `allDifferent`

Example.

For a Sudoku block, a constraint `allDifferent`(w, x, y, z) :



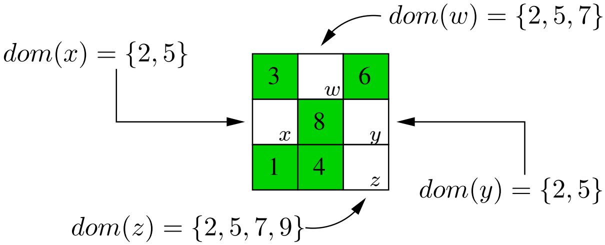
Can we filter?

Identification of Hall sets

The same constraint as previously, but variables have different domains.

Example.

For a Sudoku block, a constraint `allDifferent`(w, x, y, z) :

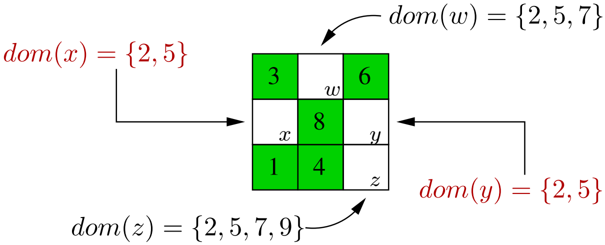


Identification of Hall sets

The same constraint as previously, but variables have different domains.

Example.

For a Sudoku block, a constraint allDifferent(w, x, y, z) :

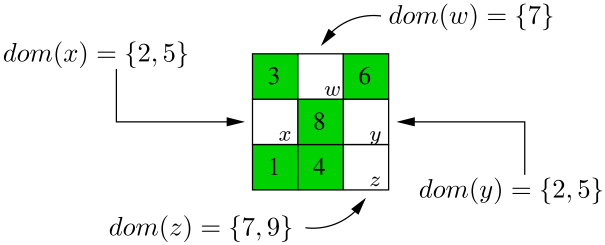


Identification of Hall sets

The same constraint as previously, but variables have different domains.

Example.

For a Sudoku block, a constraint `allDifferent`(w, x, y, z) :

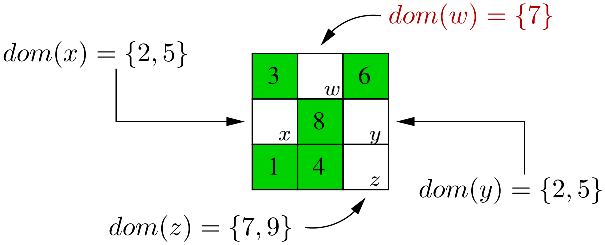


Identification of Hall sets

The same constraint as previously, but variables have different domains.

Example.

For a Sudoku block, a constraint `allDifferent`(w, x, y, z) :

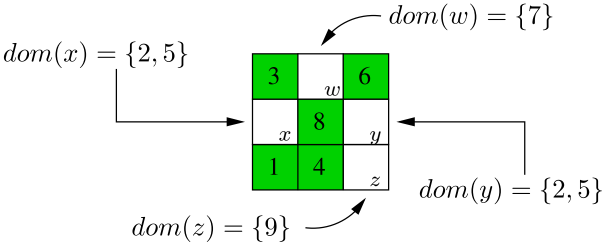


Identification of Hall sets

The same constraint as previously, but variables have different domains.

Example.

For a Sudoku block, a constraint `allDifferent`(w, x, y, z) :



AC Filtering for cardinality

Definition

A constraint `cardinality`(X, V, L, U) forces the variables in X to take their values in V with the restriction that each value v_i in V is assigned at least $L(v_i)$ times and at most $U(v_i)$ times.

Example.

Three sets:

- $Agents = \{Peter, Paul, Mary, John, Bob, Mike, Julia\}$
- $Days = \{Monday, Tuesday, ..., Sunday\}$
- $Activities = \{M(orning), D(ay), N(ight), B(ackup), O(ff)\}$.
- We want a roster that looks like:

	Mo	Tu	We	Th	Fr	Sa	Su
Peter	D	N	N	N	O	O	O
Paul	O	O	D	D	M	M	B
Mary	M	M	D	D	O	O	N
...							

AC Filtering for `cardinality`

Definition

A constraint `cardinality`(X, V, L, U) forces the variables in X to take their values in V with the restriction that each value v_i in V is assigned at least $L(v_i)$ times and at most $U(v_i)$ times.

Example.

Three sets:

- $Agents = \{Peter, Paul, Mary, John, Bob, Mike, Julia\}$
- $Days = \{Monday, Tuesday, ..., Sunday\}$
- $Activities = \{M(orning), D(ay), N(ight), B(ackup), O(ff)\}$.
- We want a roster that looks like:

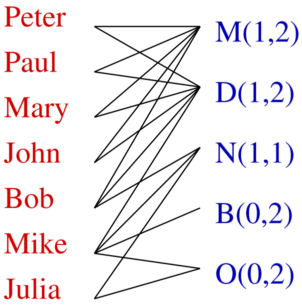
	Mo	Tu	We	Th	Fr	Sa	Su
Peter	D	N	N	N	O	O	O
Paul	O	O	D	D	M	M	B
Mary	M	M	D	D	O	O	N
...							

AC Filtering for `cardinality`

Example.

For simplicity, we only reason here on Monday. Our variables X represent the agents, and we have $\forall x \in X, dom(x) = \{M, D, N, B, O\}$.
The constraint `cardinality`(X,$\{M, D, N, B, O\}$,L,U) is such that:

- $L = \{1, 1, 1, 0, 0\}$
- $U = \{2, 2, 1, 2, 2\}$.

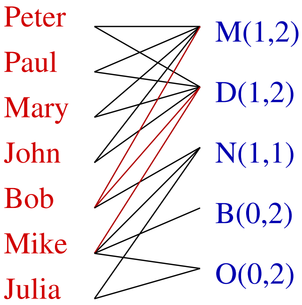


AC Filtering for `cardinality`

Example.

For simplicity, we only reason here on Monday. Our variables X represent the agents, and we have $\forall x \in X, dom(x) = \{M, D, N, B, O\}$.
The constraint `cardinality`(X,$\{M, D, N, B, O\}$,L,U) is such that:

- $L = \{1, 1, 1, 0, 0\}$
- $U = \{2, 2, 1, 2, 2\}$.

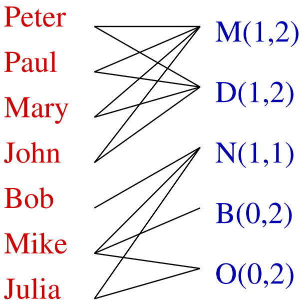


AC Filtering for cardinality

Example.

For simplicity, we only reason here on Monday. Our variables X represent the agents, and we have $\forall x \in X, dom(x) = \{M, D, N, B, O\}$.
The constraint `cardinality`$(X, \{M, D, N, B, O\}, L, U)$ is such that:

- $L = \{1, 1, 1, 0, 0\}$
- $U = \{2, 2, 1, 2, 2\}$.

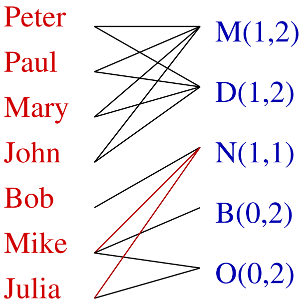


AC Filtering for `cardinality`

Example.

For simplicity, we only reason here on Monday. Our variables X represent the agents, and we have $\forall x \in X, dom(x) = \{M, D, N, B, O\}$.
The constraint `cardinality`(X,$\{M, D, N, B, O\}$,L,U) is such that:

- $L = \{1, 1, 1, 0, 0\}$
- $U = \{2, 2, 1, 2, 2\}$.

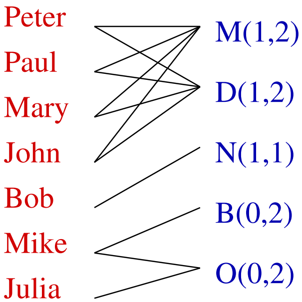


AC Filtering for `cardinality`

Example.

For simplicity, we only reason here on Monday. Our variables X represent the agents, and we have $\forall x \in X, dom(x) = \{M, D, N, B, O\}$.
The constraint `cardinality`$(X,\{M, D, N, B, O\},L,U)$ is such that:

- $L = \{1, 1, 1, 0, 0\}$
- $U = \{2, 2, 1, 2, 2\}$.



AC Filtering for `sum` : $\sum_{i=1}^{r} c_i x_i \geq L$

Domains of variables
w, x, y and z

dom			
w	x	y	z
1	1	2	2
2	2	3	3
3	3	4	4

Constraint $c_{wxyz} : w + 2x + 4y + 5z \geq 42$

Domains of variables
w, x, y et z
after AC filtering of c_{wxyz}

dom			
w	x	y	z
1	~~1~~	~~2~~	~~2~~
2	2	~~3~~	~~3~~
3	3	4	4

Complexity?

AC Filtering for `sum` : $\sum_{i=1}^{r} c_i x_i \geq L$

Domains of variables
w, x, y and z

dom			
w	x	y	z
1	1	2	2
2	2	3	3
3	3	4	4

Constraint c_{wxyz} : $w + 2x + 4y + 5z \geq 42$

Domains of variables
w, x, y et z
after AC filtering of c_{wxyz}

dom			
w	x	y	z
1	~~1~~	~~2~~	~~2~~
2	2	~~3~~	~~3~~
3	3	4	4

Complexity?

AC Filtering for sum : $\sum_{i=1}^{r} c_i x_i \geq L$

Domains of variables
w, x, y and z

dom			
w	x	y	z
1	1	2	2
2	2	3	3
3	3	4	4

Constraint c_{wxyz} : $w + 2x + 4y + 5z \geq 42$

Domains of variables
w, x, y et z
after AC filtering of c_{wxyz}

dom			
w	x	y	z
1	~~1~~	~~2~~	~~2~~
2	2	~~3~~	~~3~~
3	3	4	4

Complexity?

AC Filtering for $\texttt{sum}$: $\sum_{i=1}^{r} c_i x_i \geq L$

Domains of variables
w, x, y and z

dom			
w	x	y	z
1	1	2	2
2	2	3	3
3	3	4	4

Constraint c_{wxyz} : $w + 2x + 4y + 5z \geq 42$

Domains of variables
w, x, y et z
after AC filtering of c_{wxyz}

dom			
w	x	y	z
1	~~1~~	~~2~~	~~2~~
2	2	~~3~~	~~3~~
3	3	4	4

Complexity?

AC Filtering for $\texttt{sum}$: $\sum_{i=1}^{r} c_i x_i \neq L$

Domains of variables
w, x, y and z

dom			
w	x	y	z
1	1	1	1
		2	2

Constraint $c_{wxyz} : w + x + y + z \neq 5$

Domains of variables
w, x, y and z
after AC filtering of c_{wxyz}

dom			
w	x	y	z
1	1	1	1
		2	2

AC Filtering for $\texttt{sum}$: $\sum_{i=1}^{r} c_i x_i \neq L$

Domains of variables
w, x, y and z

dom			
w	x	y	z
1	1	1	1
		2	2

Constraint c_{wxyz} : $w + x + y + z \neq 5$

Domains of variables
w, x, y and z
after AC filtering of c_{wxyz}

dom			
w	x	y	z
1	1	1	1
		2	2

AC Filtering for sum : $\sum_{i=1}^{r} c_i x_i \neq L$

Domains of variables
w, x, y and z

dom			
w	x	y	z
1	1	1	1
		2	2

Constraint $c_{wxyz} : w + x + y + z \neq 5$

Domains of variables
w, x, y and z
after AC filtering of c_{wxyz}

dom			
w	x	y	z
1	1	1	1
		2	2

AC Filtering for `sum` : $\sum_{i=1}^{r} c_i x_i \neq L$

Domains of variables
w, x, y and z

dom			
w	x	y	z
1	1	1	1
		2	

Constraint $c_{wxyz} : w + x + y + z \neq 5$

Domains of variables
w, x, y and z
after AC filtering of c_{wxyz}

dom			
w	x	y	z
1	1	1	1
		~~2~~	

Complexity?

AC Filtering for sum : $\sum_{i=1}^{r} c_i x_i \neq L$

Domains of variables
w, x, y and z

dom			
w	x	y	z
1	1	1	1
		2	

Constraint $c_{wxyz} : w + x + y + z \neq 5$

Domains of variables
w, x, y and z
after AC filtering of c_{wxyz}

dom			
w	x	y	z
1	1	1	1
		~~2~~	

Complexity?

AC Filtering for $\texttt{sum}$: $\sum_{i=1}^{r} c_i x_i \neq L$

Domains of variables
w, x, y and z

dom			
w	x	y	z
1	1	1	1
		2	

Constraint c_{wxyz} : $w + x + y + z \neq 5$

Domains of variables
w, x, y and z
after AC filtering of c_{wxyz}

dom			
w	x	y	z
1	1	1	1
		~~2~~	

Complexity?

AC Filtering for sum : $\sum_{i=1}^{r} c_i x_i \neq L$

Domains of variables
w, x, y and z

dom			
w	x	y	z
1	1	1	1
		2	

Constraint $c_{wxyz} : w + x + y + z \neq 5$

Domains of variables
w, x, y and z
after AC filtering of c_{wxyz}

dom			
w	x	y	z
1	1	1	1
		~~2~~	

Complexity?

AC Filtering for $\texttt{sum}$: $U \geq \sum_{i=1}^{r} c_i x_i \geq L$

Domains of variables
w, x, y and z

dom			
w	x	y	z
0	0	0	0
1	1	1	1
2	2	2	2
3	3	3	3

Constraint $c_{wxyz} : 82 \geq 27w + 37x + 45y + 53z \geq 80$

Domains of variables
w, x, y and z
after AC filtering of c_{wxyz}

dom			
w	x	y	z
0	0	0	0
1	1	1	1
~~2~~	~~2~~	~~2~~	~~2~~
3	~~3~~	~~3~~	~~3~~

AC Filtering for `sum` : $U \geq \sum_{i=1}^{r} c_i x_i \geq L$

Domains of variables
w, x, y and z

dom			
w	x	y	z
0	0	0	0
1	1	1	1
2	2	2	2
3	3	3	3

Constraint c_{wxyz} : $82 \geq 27w + 37x + 45y + 53z \geq 80$

Domains of variables
w, x, y and z
after AC filtering of c_{wxyz}

dom			
w	x	y	z
0	0	0	0
1	1	1	1
~~2~~	~~2~~	~~2~~	~~2~~
3	~~3~~	~~3~~	~~3~~

AC Filtering for sum : $U \geq \sum_{i=1}^{r} c_i x_i \geq L$

Domains of variables
w, x, y and z

dom			
w	x	y	z
0	0	0	0
1	1	1	1
2	2	2	2
3	3	3	3

Constraint c_{wxyz} : $82 \geq 27w + 37x + 45y + 53z \geq 80$

Domains of variables
w, x, y and z
after AC filtering of c_{wxyz}

dom			
w	x	y	z
0	0	0	0
1	1	1	1
~~2~~	~~2~~	~~2~~	~~2~~
3	~~3~~	~~3~~	~~3~~

AC Filtering for sum : $U \geq \sum_{i=1}^{r} c_i x_i \geq L$

Possibility of using dynamic programming:

- construction of a graph (Knapsack)
- reduction of the graph
- use of a constraint `mdd` from the reduced graph

Warning.

Pseudo-polynomial Complexity $O(rU^2)$

Example.

Illustration of this approach with:

- the constraint $12 \geq 2x_1 + 3x_2 + 4x_3 + 5x_4 \geq 10$
- where the domain of each variable is $\{0, 1\}$.

AC Filtering for `sum` : $U \geq \sum_{i=1}^{r} c_i x_i \geq L$

Possibility of using dynamic programming:

- construction of a graph (Knapsack)
- reduction of the graph
- use of a constraint `mdd` from the reduced graph

Warning.

Pseudo-polynomial Complexity $O(rU^2)$

Example.

Illustration of this approach with:

- the constraint $12 \geq 2x_1 + 3x_2 + 4x_3 + 5x_4 \geq 10$
- where the domain of each variable is $\{0, 1\}$.

AC Filtering for `sum` : $U \geq \sum_{i=1}^{r} c_i x_i \geq L$

Possibility of using dynamic programming:

- construction of a graph (Knapsack)
- reduction of the graph
- use of a constraint `mdd` from the reduced graph

Warning.

Pseudo-polynomial Complexity $O(rU^2)$

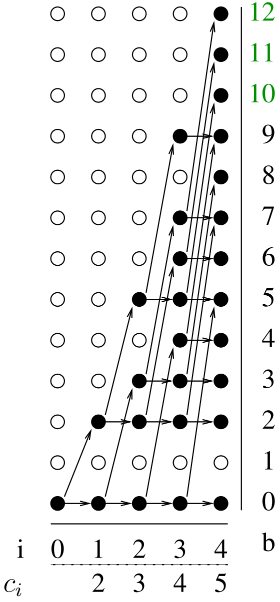
Example.

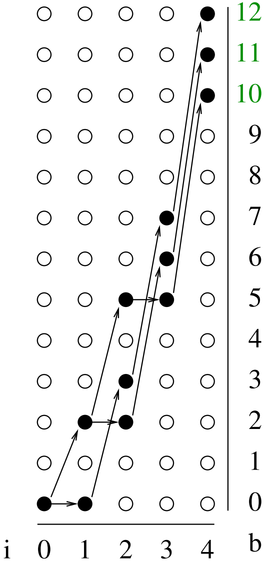
Illustration of this approach with:

- the constraint $12 \geq 2x_1 + 3x_2 + 4x_3 + 5x_4 \geq 10$
- where the domain of each variable is $\{0, 1\}$.

Example

Knapsack Graph





Reduced Knapsack Graph

AC Filtering for `or` (meta-constraint)

Constructive Disjunction

Enforcing AC on a meta-constraint `or`(c_1,c_2) can be achieved by *constructive disjunction*: for each variable x, $dom(x)$ is the union of the domains of x obtained after AC filtering on c_1 and AC filtering on c_2.

Example.

Let x be a variable such that $dom(x) = \{1, 2, 3\}$ and the meta-constraint `or`($x = 1$,$x = 2$).

AC on $x = 1$ *yields* $dom^1(x) = \{1\}$
AC on $x = 2$ *yields* $dom^2(x) = \{2\}$

AC on `or`($x = 1$,$x = 2$) reduces $dom(x)$ to $dom^1(x) \cup dom^2(x) = \{1, 2\}$

AC Filtering for `or` (meta-constraint)

Constructive Disjunction

Enforcing AC on a meta-constraint $\texttt{or}(c_1,c_2)$ can be achieved by *constructive disjunction*: for each variable x, $dom(x)$ is the union of the domains of x obtained after AC filtering on c_1 and AC filtering on c_2.

Example.

Let x be a variable such that $dom(x) = \{1, 2, 3\}$ and the meta-constraint $\texttt{or}(x = 1, x = 2)$.

AC on $x = 1$ *yields* $dom^1(x) = \{1\}$
AC on $x = 2$ *yields* $dom^2(x) = \{2\}$

AC on $\texttt{or}(x = 1, x = 2)$ reduces $dom(x)$ to $dom^1(x) \cup dom^2(x) = \{1, 2\}$

AC Filtering for and (meta-constraint)

Proposition

AC on the conjunction `and`*(c_1,c_2) is with respect to AC enforced independently on c_1 and c_2:*

- *generally stronger,*
- *equivalent when* $|scp(c_1) \cap scp(c_2)| \leq 1$

Example

Let x and y two variables such that $dom(x) = dom(y) = \{1, 2, 3\}$ and the meta-constraint `and`($x \neq y$,$x \leq y$).

- AC on $x \neq y$ as well as AC on $x \leq y$ have no effect
- AC on `and`($x \neq y$,$x \leq y$) permits to have:
  - $dom(x)$ reduced to $\{1, 2\}$
  - $dom(y)$ reduced to $\{2, 3\}$

AC Filtering for `and` (meta-constraint)

Proposition

AC on the conjunction `and`(c_1,c_2) *is with respect to AC enforced independently on* c_1 *and* c_2*:*

- *generally stronger,*
- *equivalent when* $|scp(c_1) \cap scp(c_2)| \leq 1$

Example

Let x and y two variables such that $dom(x) = dom(y) = \{1, 2, 3\}$ and the meta-constraint `and`$(x \neq y, x \leq y)$.

- AC on $x \neq y$ as well as AC on $x \leq y$ have no effect
- AC on `and`$(x \neq y, x \leq y)$ permits to have:
  - $dom(x)$ reduced to $\{1, 2\}$
  - $dom(y)$ reduced to $\{2, 3\}$

Outline

1. Filtering Domains with Constraints

2. Principle of Constraint Propagation

Constraint Propagation

Definition

A constraint network P is AC iff each constraint of P is AC.

Definition

Computing the AC-closure of a constraint network P is the fact of removing all arc-inconsistent of P (when considering any constraint of P).

May we sollicit each constraint once (for filtering) in order to compute the AC-closure?

NO because when some values are filtered out by a constraint, this can give new opportunities to other constraints to filter again.

The process that involves executing filtering operations, by solliciting constraints in turn, until a fixed point is reached is called **constraint propagation**.

Constraint Propagation

Definition

A constraint network P is AC iff each constraint of P is AC.

Definition

Computing the AC-closure of a constraint network P is the fact of removing all arc-inconsistent of P (when considering any constraint of P).

May we sollicit each constraint once (for filtering) in order to compute the AC-closure?

NO because when some values are filtered out by a constraint, this can give new opportunities to other constraints to filter again.

The process that involves executing filtering operations, by solliciting constraints in turn, until a fixed point is reached is called **constraint propagation**.

Constraint Propagation

Definition

A constraint network P is AC iff each constraint of P is AC.

Definition

Computing the AC-closure of a constraint network P is the fact of removing all arc-inconsistent of P (when considering any constraint of P).

May we sollicit each constraint once (for filtering) in order to compute the AC-closure?

NO because when some values are filtered out by a constraint, this can give new opportunities to other constraints to filter again.

The process that involves executing filtering operations, by solliciting constraints in turn, until a fixed point is reached is called **constraint propagation**.

Constraint Propagation

Definition
A constraint network P is AC iff each constraint of P is AC.

Definition
Computing the AC-closure of a constraint network P is the fact of removing all arc-inconsistent of P (when considering any constraint of P).

May we sollicit each constraint once (for filtering) in order to compute the AC-closure?

NO because when some values are filtered out by a constraint, this can give new opportunities to other constraints to filter again.

The process that involves executing filtering operations, by solliciting constraints in turn, until a fixed point is reached is called **constraint propagation**.

Constraint Propagation

Definition

A constraint network P is AC iff each constraint of P is AC.

Definition

Computing the AC-closure of a constraint network P is the fact of removing all arc-inconsistent of P (when considering any constraint of P).

May we sollicit each constraint once (for filtering) in order to compute the AC-closure?

NO because when some values are filtered out by a constraint, this can give new opportunities to other constraints to filter again.

The process that involves executing filtering operations, by solliciting constraints in turn, until a fixed point is reached is called **constraint propagation**.

Constraint Propagation Algorithm

Algorithm 4: constraintPropagationOn(P: CN): Boolean

$Q \leftarrow ctrs(P)$
while $Q \neq \emptyset$ **do**
  pick and delete c from Q
  $X_{evt} \leftarrow c.filter()$ // X_{evt} `denotes the set of variables with reduced domains (after filtering by means of` c`)`
  if $\exists x \in X_{evt}$ *such that* $dom(x) = \emptyset$ **then**
    return *false* // `global inconsistency detected`
  foreach $c' \in ctrs(P)$ *such that* $c' \neq c$ *and* $X_{evt} \cap scp(c') \neq \emptyset$ **do**
    add c' to Q
return *true*

Remark.

If each call $c.filter()$ enforces AC on c, then the algorithm computes the AC-closure of P.

Constraint Propagation Algorithm

Algorithm 5: constraintPropagationOn(P: CN): Boolean

$Q \leftarrow ctrs(P)$
while $Q \neq \emptyset$ **do**
  pick and delete c from Q
  $X_{evt} \leftarrow c.filter()$ `//` X_{evt} `denotes the set of variables with reduced domains (after filtering by means of` c`)`
  if $\exists x \in X_{evt}$ *such that* $dom(x) = \emptyset$ **then**
    return *false* `// global inconsistency detected`
  foreach $c' \in ctrs(P)$ *such that* $c' \neq c$ *and* $X_{evt} \cap scp(c') \neq \emptyset$ **do**
    add c' to Q
return *true*

Remark.

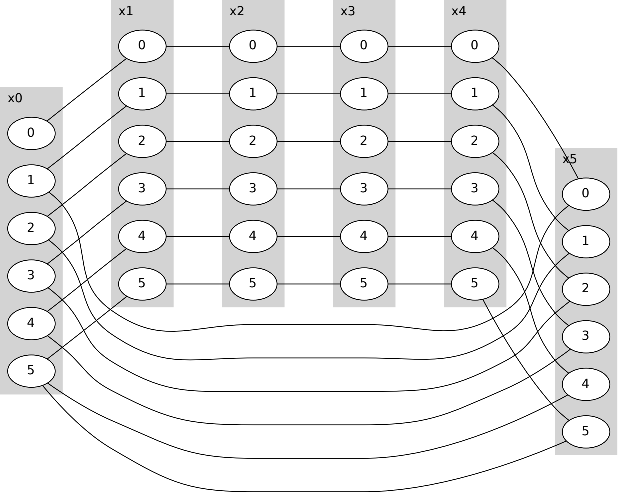
If each call $c.filter()$ enforces AC on c, then the algorithm computes the AC-closure of P.

Domino Problem

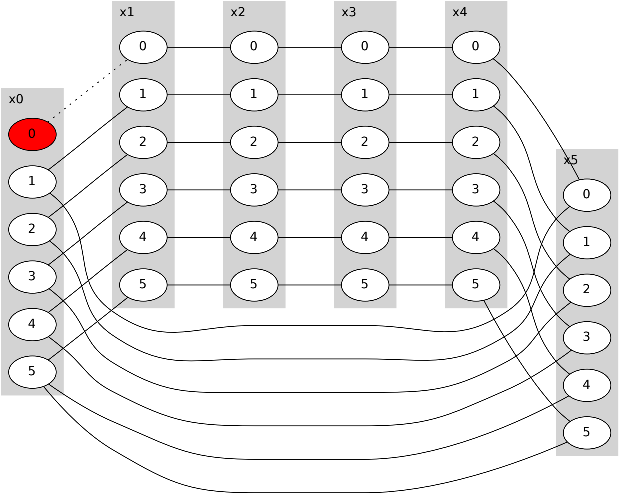
The instance `domino-6` is represented by the following CN P:

- $vars(P) = \{$
  x_0 with $dom(x_0) = \{0, 1, 2, 3, 4, 5\}$,
  x_1 with $dom(x_1) = \{0, 1, 2, 3, 4, 5\}$,
  x_2 with $dom(x_2) = \{0, 1, 2, 3, 4, 5\}$,
  x_3 with $dom(x_3) = \{0, 1, 2, 3, 4, 5\}$,
  x_4 with $dom(x_4) = \{0, 1, 2, 3, 4, 5\}$,
  x_5 with $dom(x_5) = \{0, 1, 2, 3, 4, 5\}$
  $\}$
- $ctrs(P) = \{$
  $x_0 = x_1$,
  $x_1 = x_2$,
  $x_2 = x_3$,
  $x_3 = x_4$,
  $x_4 = x_5$,
  $(x_0 = x_5 + 1 \land x_0 < 5) \lor (x_0 = x_5 \land x_0 = 5)$
  $\}$

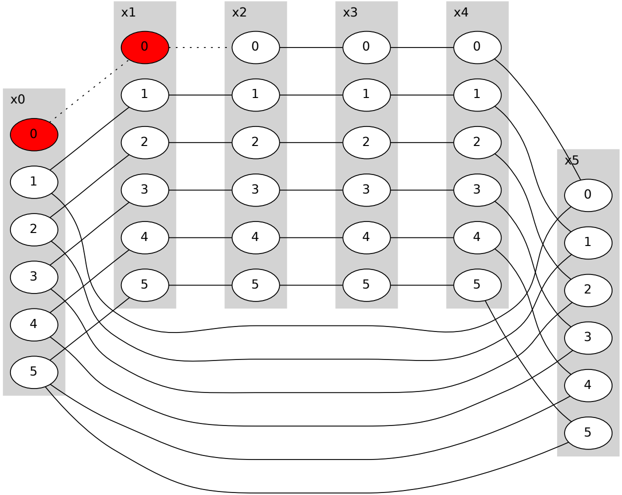
Constraint Propagation on domino-6



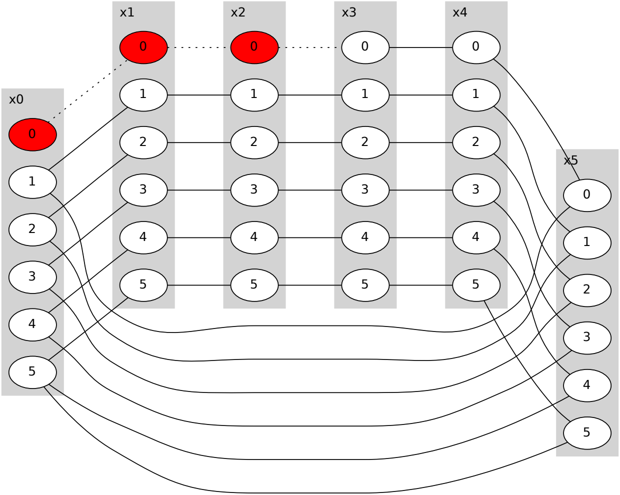
Constraint Propagation on `domino-6`



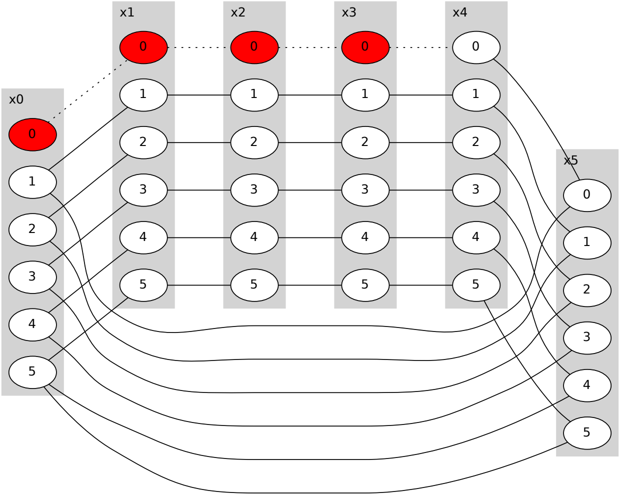
Constraint Propagation on `domino-6`



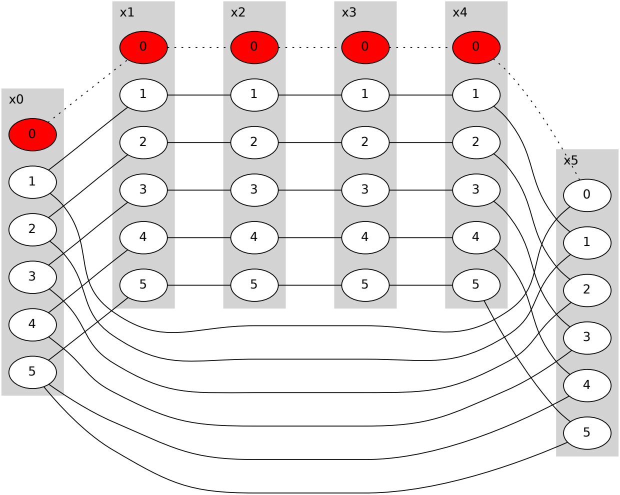
Constraint Propagation on domino-6



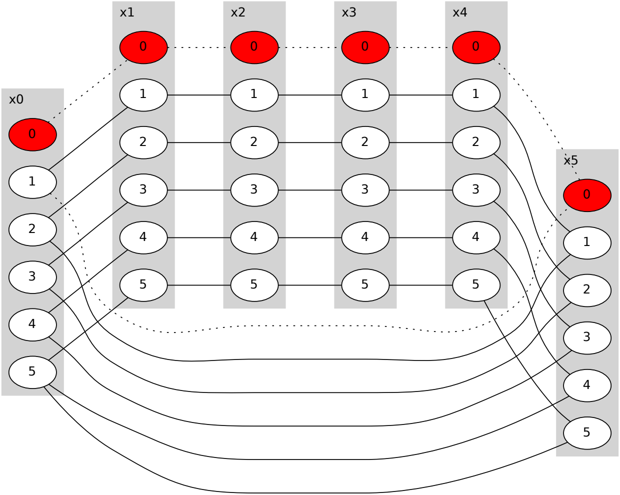
Constraint Propagation on domino-6



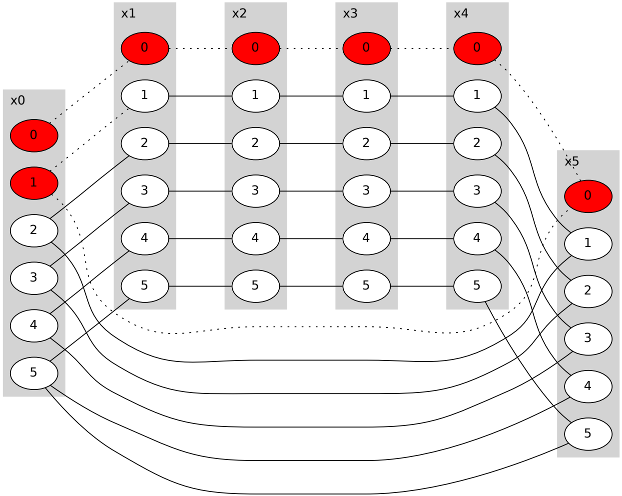
Constraint Propagation on `domino-6`



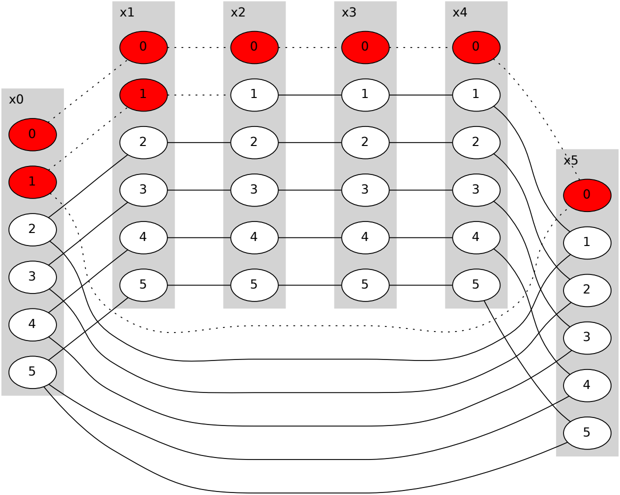
Constraint Propagation on domino-6



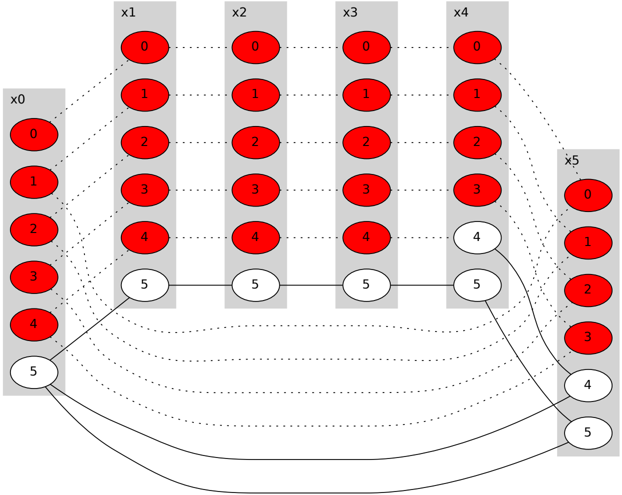
Constraint Propagation on `domino-6`



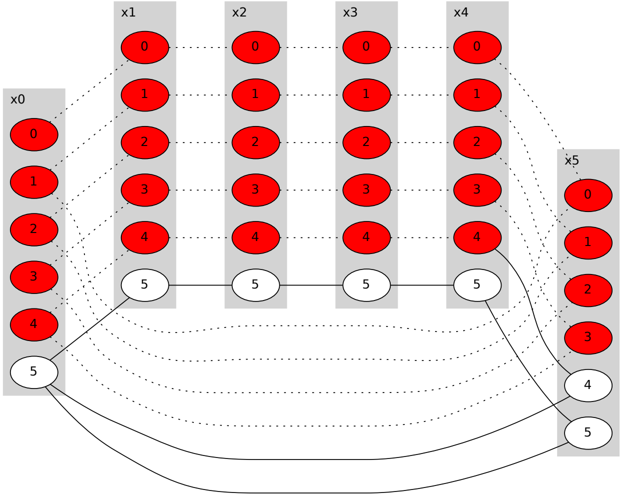
Constraint Propagation on `domino-6`



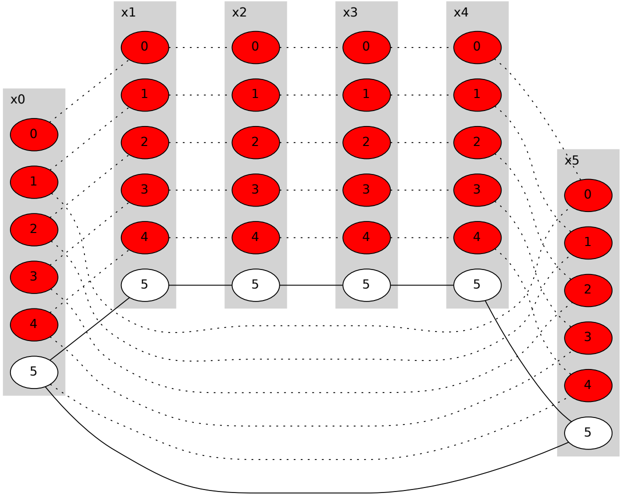
Constraint Propagation on `domino-6`



Constraint Propagation on `domino-6`



Constraint Propagation on `domino-6`



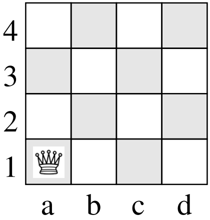
Constraint Propagation on queens-4

For the 4-queens instance, we have:

- $vars(P) = \{$
  x_a with $dom(x_a) = \{1, 2, 3, 4\}$,
  x_b with $dom(x_b) = \{1, 2, 3, 4\}$,
  x_c with $dom(x_c) = \{1, 2, 3, 4\}$,
  x_d with $dom(x_d) = \{1, 2, 3, 4\}$
  $\}$
- $ctrs(P) = \{$
  $x_a \neq x_b \wedge |x_a - x_b| \neq 1$,
  $x_a \neq x_c \wedge |x_a - x_c| \neq 2$,
  $x_a \neq x_d \wedge |x_a - x_d| \neq 3$,
  $x_b \neq x_c \wedge |x_b - x_c| \neq 1$,
  $x_b \neq x_d \wedge |x_b - x_d| \neq 2$,
  $x_c \neq x_d \wedge |x_c - x_d| \neq 1$
  $\}$

Exercice

After taking the decision $x_a = 1$, what is the AC-closure of P?



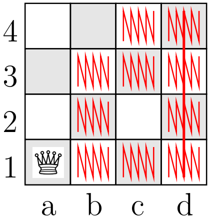
Constraint Propagation on queens-4

For the 4-queens instance, we have:

- $vars(P) = \{$
  x_a with $dom(x_a) = \{1, 2, 3, 4\}$,
  x_b with $dom(x_b) = \{1, 2, 3, 4\}$,
  x_c with $dom(x_c) = \{1, 2, 3, 4\}$,
  x_d with $dom(x_d) = \{1, 2, 3, 4\}$
  $\}$
- $ctrs(P) = \{$
  $x_a \neq x_b \wedge |x_a - x_b| \neq 1$,
  $x_a \neq x_c \wedge |x_a - x_c| \neq 2$,
  $x_a \neq x_d \wedge |x_a - x_d| \neq 3$,
  $x_b \neq x_c \wedge |x_b - x_c| \neq 1$,
  $x_b \neq x_d \wedge |x_b - x_d| \neq 2$,
  $x_c \neq x_d \wedge |x_c - x_d| \neq 1$
  $\}$

Exercice

After taking the decision $x_a = 1$, the AC-closure of P is:



Exercice

Let P be the following CN:

- $vars(P) = \{$
  x_1 with $dom(x_1) = \{1, 2, 3\}$,
  x_2 with $dom(x_2) = \{1, 2, 3\}$,
  x_3 with $dom(x_3) = \{1, 2, 3\}$,
  x_4 with $dom(x_4) = \{1, 2, 3\}$
  $\}$
- $ctrs(P) = \{$
  $x_1 \neq x_2$,
  $x_2 + x_3 \leq x_1$,
  $x_2 + x_4 \geq 2 * x_1$,
  $\}$

Simulate the process of constraint propagation on P (that is to say, compute the AC-closure of P).