Topic 3: Consistency[1]
(Version of 14th September 2017)

Pierre Flener

ASTRA Research Group
on Combinatorial Optimisation
Uppsala University
Sweden

Course 1DL441:
Combinatorial Optimisation using
Constraint Programming

[1]Based also on some material by Christian Schulte and Yves Deville

Outline

Outline

Constraint Problems

Definition (Constraint problem)

A constraint satisfaction problem (CSP) is $\langle V, D, C \rangle$ where:

- $V = [v_1, \ldots, v_m]$ is a finite sequence of variables, which are often called decision variables.
- $D = [D_1, \ldots, D_m]$ is a finite sequence of domains, which are sets of possible values for the variables.
- $C = \{c_1, \ldots, c_p\}$ is a finite set of constraints on the variables, a constraint $\gamma(v_{i_1}, \ldots, v_{i_q})$ having arity q. We often assume $i_j = j$, without loss of generality.

A constrained optimisation problem (COP) is $\langle V, D, C, f \rangle$:

- The triple $\langle V, D, C \rangle$ is a CSP.
- f is a function from $D_1 \times \cdots \times D_m$ to $\mathbb{R}$ or $\mathbb{N}$, called the objective function, which is here to be minimised, without loss of generality.

More on problems:

- Without loss of generality, we often simplify notation by requiring that all variables initially have the same domain U, called the universe: $D_1 = \cdots = D_m = U$. We then refer to a triple $\langle V, U, C \rangle$ as a CSP, and to a quadruple $\langle V, U, C, f \rangle$ as a COP.

- In this course, we focus on finite domains, and thus also refer to a CSP or COP as a combinatorial problem.

- We distinguish a problem from its instances, defined by instance data. Example: n-Queens vs 8-Queens. Some problems, such as the grocery problem, have only one instance.

- Sometimes, we refer to a single constraint as a CSP.

Stores and Solutions

Definition (Store)

The store of a CP solver is a function mapping each decision variable of a CSP or COP to its current domain.

Example

The function $\{x \mapsto \{1,2\}\,,\; y \mapsto \{2,3\}\}$ is a store.

Definition (Assigned)

A decision variable x is assigned (or fixed) under store s iff its domain under s is a singleton set: $|s(x)| = 1$.

Notation:

If the name, say s, of the current store is irrelevant, then we denote the domain $s(x)$ of a decision variable x by $\mathrm{dom}(x)$.

Definition (Solution store)

A store s is a solution store to a constraint $c = \gamma(x_1, \ldots, x_q)$ iff all domains have single values forming a solution to c: $s(x_i) = \{d_i\}$ for all $i \in [1, q]$, and $\langle d_1, \ldots, d_q \rangle$ is solution to c.

Example

The store $\{x \mapsto \{3\},\ y \mapsto \{4\}\}$ is a solution store to $x \leq y$.

Definition (Solution membership in a store)

A solution $\langle d_1, \ldots, d_q \rangle$ to a constraint $\gamma(x_1, \ldots, x_q)$ is in (denoted $\in$) a store s iff every value belongs to the domain of the corresponding variable: $d_i \in s(x_i)$, for all $i \in [1, q]$.

Example

The solution $\langle 3, 4 \rangle$ to the constraint $x \leq y$ is in the store $\{x \mapsto \{1, 3\},\ y \mapsto \{2, 4\},\ z \mapsto \{5, 6\}\}$.

Outline

UPPSALA UNIVERSITET

Value Consistency

Example (Value consistency for DISTINCT)

If a variable is assigned, then its value does not appear in the domains of all the other variables of the constraint.
Consider DISTINCT($\{x, y, z\}$):

- Store $s = \{x, y \mapsto \{1, 2\}, z \mapsto \{3\}\}$ is value consistent.
- Store $s = \{x, y, z \mapsto \{1, 2\}\}$ is value consistent, hence search is needed to show that there is no solution in s.
- Store $s = \{x, y \mapsto \{1, 2\},\ z \mapsto \{1, 2, 3\}\}$ is value consistent, hence search is needed to show that there are two solutions in s, both with $z = 3$.

Enforcing value consistency on DISTINCT($\{x_1, \dots, x_q\}$) is known as naïve DISTINCT, and takes $\mathcal{O}(q)$ time:

- Store $\{w, x, y, z \mapsto \{1, 2, 3\}\}$ is contracted upon $w = 3$ to the store $\{w \mapsto \{3\},\ x, y, z \mapsto \{1, 2\}\}$.

Enforcing value consistency:
To enforce value consistency for a constraint $\gamma(\cdots)$: whenever a decision variable is assigned a value, any impossible values according to $\gamma(\cdots)$ are removed from the domains of its other decision variables.

More about value consistency:
In the literature, value consistency (denoted below by VC) is also known as forward-checking consistency (FCC).

Consistency in General

- We will now study other consistencies.
- The enforcing (or achieving) of some consistency is called propagation and is performed by an algorithm called a propagator: ☞ in-depth discussion in Topic 4.
- Constraint predicates are often equipped with propagators for multiple consistencies, one being the default, each having different time & space complexity. Typically, but not always, a propagator takes time polynomial in the number of its decision variables.
- The modeller must make experiments for each constraint in order to choose a suitable consistency for the problem at hand and typical instances thereof.

UPPSALA UNIVERSITET

Outline

Domain Consistency

Definition (Domain consistency)

A store s is domain consistent for a constraint $\gamma(\cdots)$ iff for each decision variable v and each value in its domain $s(v)$, there exist values in the domains of the other variables such that all these values form a solution to $\gamma(\cdots)$.

Example (Domain consistency for DISTINCT($\{x, y, z\}$))

- Store $s = \{x, y, z \mapsto \{1, 2\}\}$ is domain inconsistent. Store $s' = \{x, y, z \mapsto \emptyset\}$ is domain consistent, hence no search is needed to show that there is no solution in s'.
- $\{x, y \mapsto \{1, 2\},\ z \mapsto \{1, 2, 3\}\}$ is domain inconsistent. $\{x, y \mapsto \{1, 2\},\ z \mapsto \{3\}\}$ is domain consistent, so no search is needed to show that $z = 3$ in all solutions.

☞ Topic 10: Enforcing Domain Consistency for DISTINCT.

Example (Domain consistency for $x \neq y, y \neq z, z \neq x$)

- $\{x, y, z \mapsto \{1, 2\}\}$ is domain consistent for all three constraints, hence search is needed to show that there is no solution in this store.
- $\{x, y \mapsto \{1, 2\},\ z \mapsto \{1, 2, 3\}\}$ is domain consistent, hence search is needed to show $z = 3$ in all solutions.

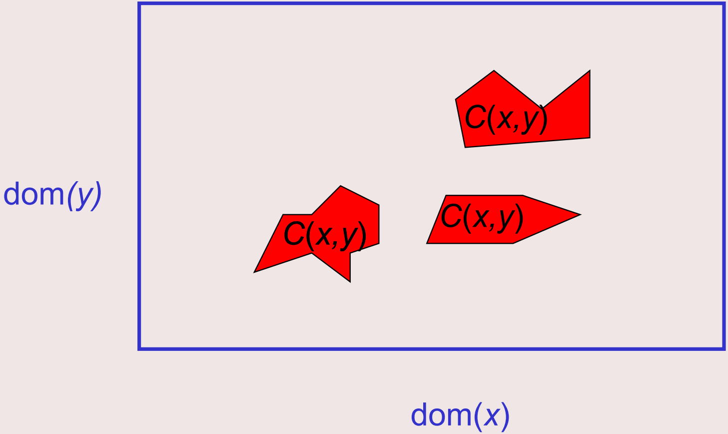
Decomposing constraint DISTINCT($\{x_1, \ldots, x_q\}$) into $\frac{q \cdot (q-1)}{2}$ constraints $x_i \neq x_j$ ($1 \leq i < j \leq q$) yields VC for DISTINCT and requires $\mathcal{O}\left(q^2\right)$ space. ☞ Topic 6: Global Constraints.

Example (Domain consistency for $x = 3 \cdot y + 5 \cdot z$)

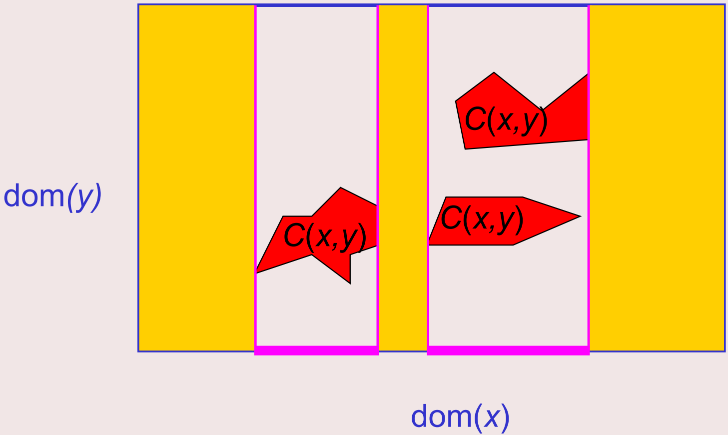
- Only the solutions $\langle 3, 1, 0\rangle$, $\langle 5, 0, 1\rangle$, and $\langle 6, 2, 0\rangle$ are in $\{x \mapsto \{2, \ldots, 7\},\ y \mapsto \{0, 1, 2\},\ z \mapsto \{-1, \ldots, 2\}\}$.
- Hence $\{x \mapsto \{3, 5, 6\},\ y \mapsto \{0, 1, 2\},\ z \mapsto \{0, 1\}\}$ is domain consistent. But we have lost the solutions!

☞ CP = reasoning with sets of (at least all) possible values!

Geometric intuition (pictures: © Yves Deville)

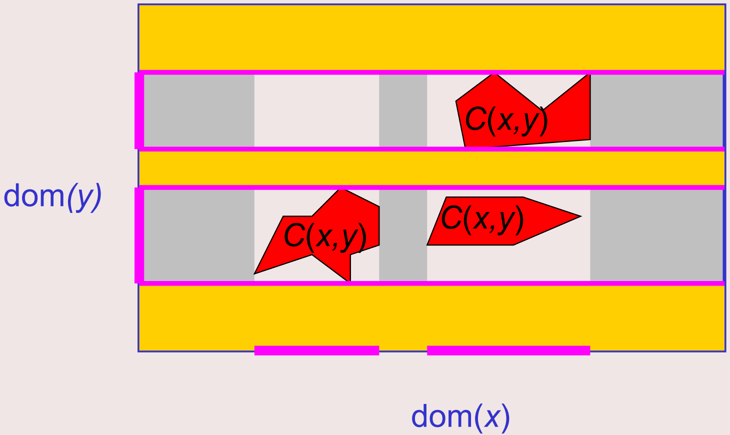


Geometric intuition (pictures: © Yves Deville)



Contracting the domain of x

Geometric intuition (pictures: © Yves Deville)



Contracting the domain of *y*

More about domain consistency:

- In the literature, domain consistency (denoted by DC) is also known as generalised arc consistency (GAC) or hyper-arc consistency (HAC), and as arc consistency (AC) in the case of binary (arity 2) constraints.
- DC is the strongest consistency, and thus implies VC for instance, but enforcing it is sometimes prohibitively expensive, for instance on linear equality constraints.
- A naïve way to enforce DC for a constraint is first to compute its solutions and then to lose them by projection for each variable: this is impractical!
  ☞ It is often possible to exploit the combinatorial structure of a constraint in order to enforce DC much faster: use global constraints (see Topics 6, 10, & 11).

Outline

Bounds Consistency

Example (Consistency for $2 \cdot x = y$)

Consider the store $s = \{x \mapsto \{1, 2, 3\}\ ,\ y \mapsto \{1, 2, 3, 4\}\}$:

- Enforcing DC contracts s to $\{x \mapsto \{1, 2\}\ ,\ y \mapsto \{2, 4\}\}$.
- But *Gecode* contracts s to $\{x \mapsto \{1, 2\}\ ,\ y \mapsto \{2, 3, 4\}\}$!

Definition (Bounds($\mathbb{Z}$) and bounds($\mathbb{R}$) consistencies)

A store s is bounds($\mathbb{Z}$) consistent for a constraint $\gamma(\cdots)$ iff for each decision variable v and the lower & upper bounds of its domain $s(v)$, there exist values between the bounds of the domains of the other variables such that all these values form an integer solution to $\gamma(\cdots)$.

Similarly for a store being bounds($\mathbb{R}$) consistent.

Definition (Bounds(D) consistency)

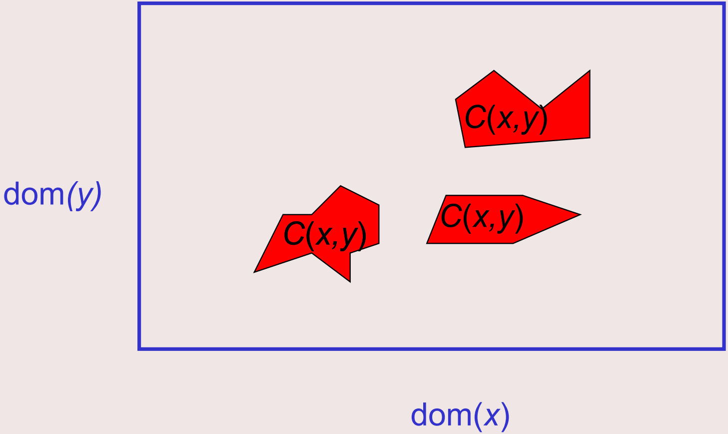
A store s is bounds(D) consistent for a constraint $\gamma(\cdots)$ iff for each decision variable v and the lower & upper bounds of its domain $s(v)$, there exist values in the domains of the other variables such that all values form a solution to $\gamma(\cdots)$.

Example (Bounds consistencies for $\max(x, y) = z$)

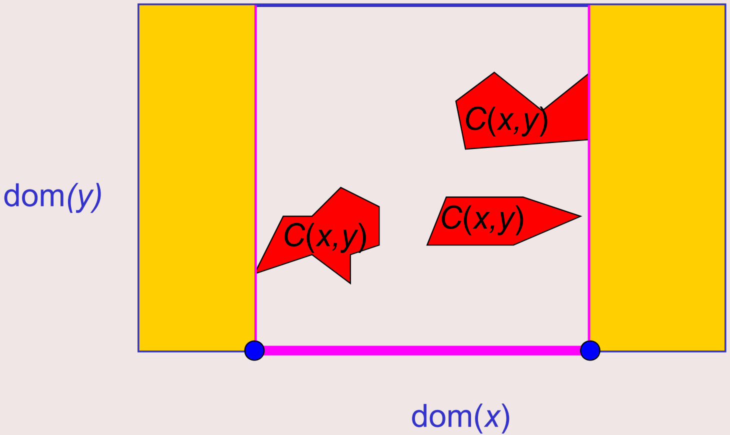
Consider $s = \{x \mapsto \{2, 3, 5\},\ y \mapsto \{3, 4, 6\},\ z \mapsto \{4, 6\}\}$:

- Enforcing bounds($\mathbb{Z}$) or bounds($\mathbb{R}$) consistency leaves s unchanged.
- Enforcing bounds(D) consistency contracts s to $\{x \mapsto \{2, 3, 5\},\ y, z \mapsto \{4, 6\}\}$.

Geometric intuition (pictures: © Yves Deville)



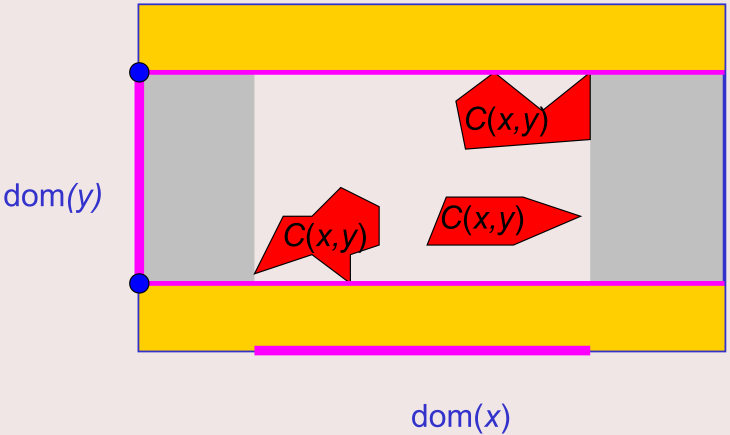
Geometric intuition (pictures: © Yves Deville)



Contracting the domain of x

UPPSALA UNIVERSITET

Geometric intuition (pictures: © Yves Deville)



Contracting the domain of y

More about bounds consistencies:

In the literature, bounds($\mathbb{R}$) consistency, denoted below by BC($\mathbb{R}$), is also known as interval consistency. By default, *Gecode* enforces BC($\mathbb{R}$) on arithmetic constraints. Note:

$$\text{DC} \Rightarrow \text{BC}(\text{D}) \Rightarrow \text{VC}$$

$$\text{BC}(\text{D}) \Rightarrow \text{BC}(\mathbb{Z}) \Rightarrow \text{BC}(\mathbb{R})$$

Example (Consistency for SEND + MORE = MONEY)

Enforcing DC on both DISTINCT($\cdots$) and the linear equality suffices to solve the problem, without search!
However, this is not faster than search interleaved with enforcing DC on DISTINCT($\cdots$) and BC($\mathbb{R}$) on the linear equality, as the problem instance is too small.

Outline

More about Consistency

Terminology:

The existentially quantified values in the definitions of DC and BC(·) are called supports (or witnesses). If at least one support exists for a considered value *d* of a universally quantified decision variable *v* in those definitions, then *d* is said to be supported, else *d* is said to be unsupported.

Other consistencies:

- Not all propagators enforce VC, BC(·), or DC, which have simple definitions: there are many useful but unnamed consistencies that can be enforced.
- A pragmatic approach is often taken, contracting domains as much as possible at a reasonable time and space complexity.

Complexity of Consistencies

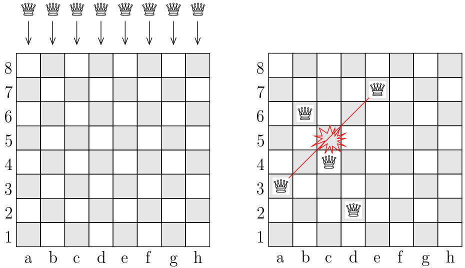
Example (DISTINCT($\{x_1, \ldots, x_q\}$))

- Value consistency: $\mathcal{O}(q)$ time
- Bounds consistency: $\mathcal{O}(q \cdot \lg q)$ time; often $\mathcal{O}(q)$ time
- Domain consistency: $\mathcal{O}(m \cdot \sqrt{q})$ time, $\mathcal{O}(m \cdot q)$ space, for $m \geq q$ domain values

Example (Linear Arithmetic on q decision variables)

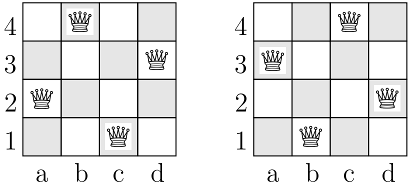
- Value consistency (useless): $\mathcal{O}(q)$ time
- Bounds consistency: $\mathcal{O}(q)$ time
- Domain consistency: exponential time (as NP-hard) for equality ($=$), but no higher time complexity than BC($\mathbb{R}$) for disequality ($\neq$) and inequality ($<, \leq, \geq, >$)

n-Queens Revisited (pics: © Ch. Lecoutre)



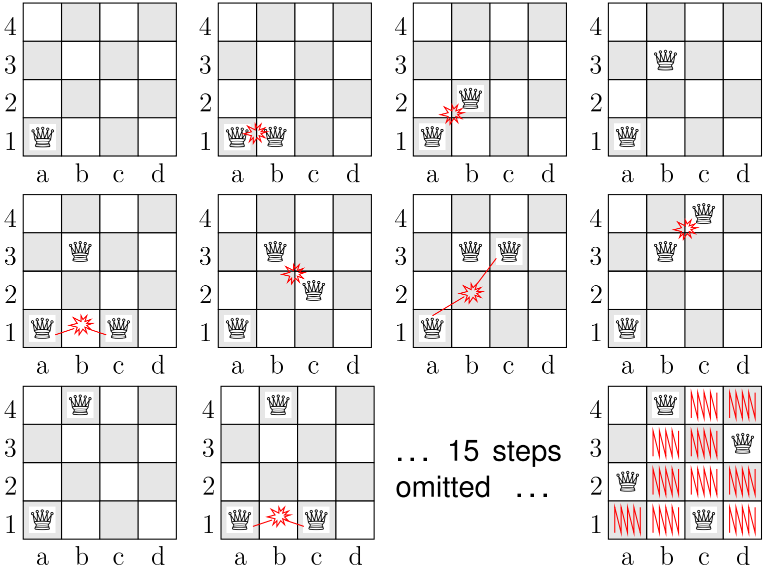
$$\text{DISTINCT}(\{r_a, r_b, .., r_h\}), \text{DISTINCT}(\{|r_a - 1|, |r_b - 2|, \ldots, |r_h - 8|\})$$

The two solutions to the 4-queens instance:

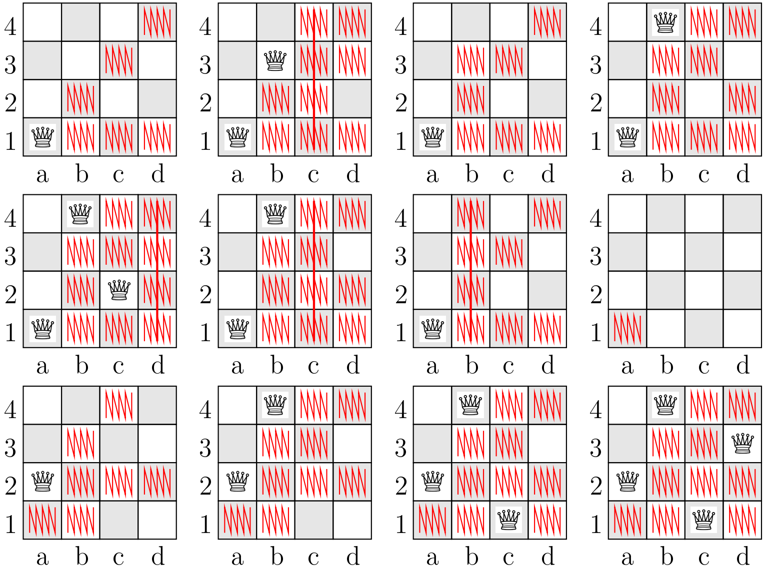


UPPSALA UNIVERSITET

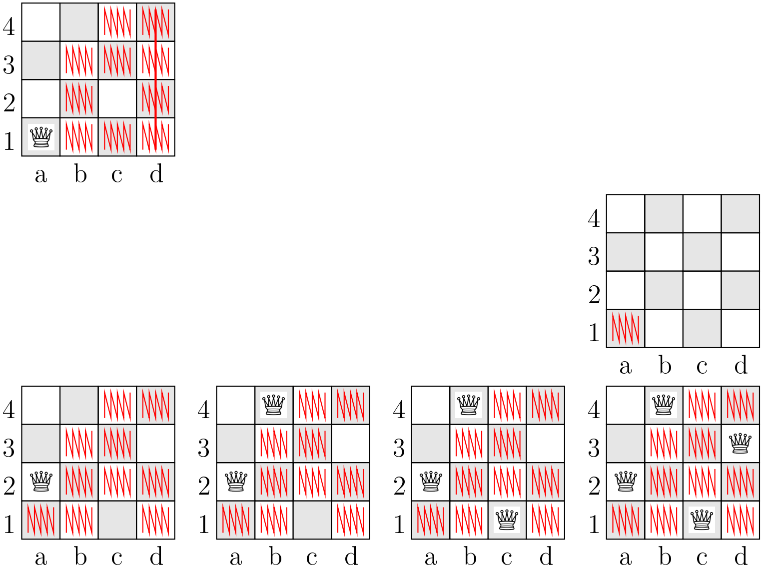
4-Queens: Backtracking Search (BT)



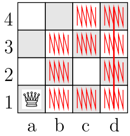
4-Queens: BT + Value Consistency (VC)

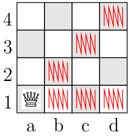


4-Queens: BT + Domain Consistency (DC)



4-Queens: BT + DC (versus BT + VC)

Why 

under DC, versus 

under VC?

Assume the search guess $r_a = 1$ is tried:

1. The DISTINCT($\{r_a, r_b, r_c, r_d\}$) row constraint propagates to $\{r_a \mapsto \{1\},\ r_b, r_c, r_d \mapsto \{2,3,4\}\}$.
2. The DISTINCT($\{|r_a - 1|, |r_b - 2|, |r_c - 3|, |r_d - 4|\}$) diagonal constraint first propagates, like under VC, to $\{r_a \mapsto \{1\},\ r_b \mapsto \{3,4\},\ r_c \mapsto \{2,4\},\ r_d \mapsto \{2,3\}\}$.
3. The previous propagator also notices that r_b cannot be 3 as the domain of r_c would then be wiped out; etc. This would not happen with two diagonal constraints!

VC only detects the conflicts between the just assigned variable and the remaining variables, but DC also detects the conflicts between the remaining variables.

Outline

Orders

Definition (Strict partial order)

A strict partial order is a pair $\langle X, \prec \rangle$, where X is a set over which the binary relation $\prec$ is irreflexive ($\forall x \in X : x \not\prec x$) and transitive ($\forall x, y, z \in X : x \prec y \wedge y \prec z \Rightarrow x \prec z$).

Definition (Well-founded order)

A well-founded order is a strict partial order $\langle X, \prec \rangle$ in which there is no infinite decreasing sequence $\cdots \prec x_3 \prec x_2 \prec x_1$.

Definition (Lexicographic order)

Given two well-founded orders $\langle X, \prec_X \rangle$ and $\langle Y, \prec_Y \rangle$, the lexicographic order $\langle X \times Y, \prec_{\text{lex}} \rangle$ is well-founded, where $\langle x_1, y_1 \rangle \prec_{\text{lex}} \langle x_2, y_2 \rangle$ iff either $x_1 \prec_X x_2$ or $x_1 = x_2 \wedge y_1 \prec_Y y_2$. Similarly for composing more than two orders.

Functions

Definition (Fixpoint)

A fixpoint of a function $f\colon X \to X$ is an element $x \in X$ that does not change under f, that is $f(x) = x$.

Idempotent functions compute fixpoints:

Definition (Idempotency)

A function f is idempotent iff it is equal to its composition with itself: $\forall x : f(f(x)) = f(x)$.