Topic 13: Consistency [1]

(Version of 20th November 2020)

Pierre Flener

Optimisation Group
Department of Information Technology
Uppsala University
Sweden

Course 1DL441:
Combinatorial Optimisation and Constraint Programming,
whose part 1 is Course 1DL451:
Modelling for Combinatorial Optimisation

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

Outline

1. Definitions
2. Value Consistency
3. Domain Consistency
4. Bounds Consistency
5. Consistency and Backtracking

Outline

Constraint Problems

Definitions

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: the set of possible values for v_i is D_i, for all $i \in 1..m$.
- $C = \{c_1, \ldots, c_p\}$ is a finite set of constraints on the variables. A constraint $\gamma(v_{i_1}, \ldots, v_{i_q})$ is a relation, of arity q. We 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, as above.
- $f(v_1, \ldots, v_m)$ is a function from $D_1 \times \cdots \times D_m$ to $\mathbb{R}$ or $\mathbb{Z}$, called the objective function, the objective here being to minimise it, without loss of generality.

More on problems:

- Without loss of generality, we often simplify notation by here 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.

- We here focus on discrete 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.

Stores and Solutions

Definition

The store of a CP solver with systematic search maps each decision variable v of a CSP or COP to its current domain, which is initialised to the domain of v in the CSP or COP.

Example

The function $\{x \mapsto \{1,7\},\ y \mapsto \{2,5\}\}$ is a store, where the current domain $\{1,7\}$ of x is $\{x \mapsto \{1,7\},\ y \mapsto \{2,5\}\}(x)$.

Definitions

A variable v is fixed in store s iff its domain is a singleton: $|s(v)| = 1$. A store s is fixed iff all its variables are fixed in s.

Notation:

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

Definition

A store s is a solution store to a constraint $c = \gamma(v_1, \ldots, v_q)$ if and only if s is fixed and denotes a solution to c: we have $s(v_i) = \{d_i\}$ for all $i \in 1..q$, and $\langle d_1, \ldots, d_q\rangle \in c$.

Example

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

Definition

A solution $\langle d_1, \ldots, d_q\rangle$ to a constraint $\gamma(v_1, \ldots, v_q)$ is in (also denoted $\in$) a store s iff every value belongs to the domain of the corresponding variable: $d_i \in s(v_i)$ for all $i \in 1..q$.

Example

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

UPPSALA UNIVERSITET

The purpose of a solver with systematic search is to find eventually and provably each (provably optimal) solution:

Definitions

For variables $V = [v_1, \ldots, v_m]$, domains $D = [D_1, \ldots, D_m]$, and a store s such that $s(v_i) = \{d_i\}$ for all $i \in 1..m$:

- The store s is a solution store to a CSP $\langle V, D, C\rangle$ if and only if s is a solution store to each constraint in C and $d_i \in D_i$ for all $i \in 1..m$.
- The store s is a solution store to a COP $\langle V, D, C, f\rangle$ if and only if s is a solution store to the CSP $\langle V, D, C\rangle$ and the objective value of s, namely $f(d_1, \ldots, d_m)$, is minimal, that is less than or equal to the objective value of every other solution store to $\langle V, D, C\rangle$.

Definition

A CP solver with systematic search is correct if and only if it finds each solution store to any CSP or COP exactly once.

Overview

Definition

A consistency is the targeted characterisation of the domain values kept in the store by a propagator for a constraint, but correctness of the solver must not depend on enforcing it.

We distinguish:

- Value consistency (VC) is quite weak.
- Domain consistency (DC) is very strong.
- Bounds consistency (BC) is between VC and DC.

We now discuss VC, DC, and various flavours of BC.

UPPSALA UNIVERSITET

Outline

Value Consistency

Example (Value consistency for `distinct`)

If a variable becomes fixed, then its value does not appear in the domains of all the other variables of the constraint.

Consider `distinct`$([x, y, z])$:

- Store $s = \{x, y \mapsto \{1, 2\}, z \mapsto \{5\}\}$ 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 $s = \{x, y \mapsto \{1, 2\},\ z \mapsto \{1, 2, 5\}\}$ is value consistent, hence search is needed to show that there are two solutions in s, both with $z = 5$.

Enforcing value consistency for `distinct`$([v_1, \ldots, v_q])$ is known as naïve `distinct`, and takes $\mathcal{O}(q)$ time:

- Store $\{w, x, y, z \mapsto \{1, 2, 5\}\}$ is contracted upon $w = 5$ to the store $\{w \mapsto \{5\},\ x, y, z \mapsto \{1, 2\}\}$.

Enforcing value consistency, in general now:

To enforce value consistency for a constraint $\gamma(\cdots)$:
whenever a decision variable becomes fixed,
any impossible values according to the semantics of $\gamma(\cdots)$
are deleted 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).

UPPSALA UNIVERSITET

Definitions

Value Consistency

Domain Consistency

Bounds Consistency

Consistency and Backtracking

Outline

Domain Consistency

Definition

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 & `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'.
- $\{x, y \mapsto \{1, 2\}\,,\ z \mapsto \{1, 2, 5\}\}$ is domain inconsistent. $\{x, y \mapsto \{1, 2\}\,,\ z \mapsto \{5\}\}$ is domain consistent, so no search is needed to show that $z = 5$ in all solutions.

☞ See `distinct` propagator in Topic 16: Propagators.

UPPSALA UNIVERSITET

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.
- $\{x, y \mapsto \{1,2\},\ z \mapsto \{1,2,5\}\}$ is domain consistent, hence search is needed to show $z = 5$ in all solutions.

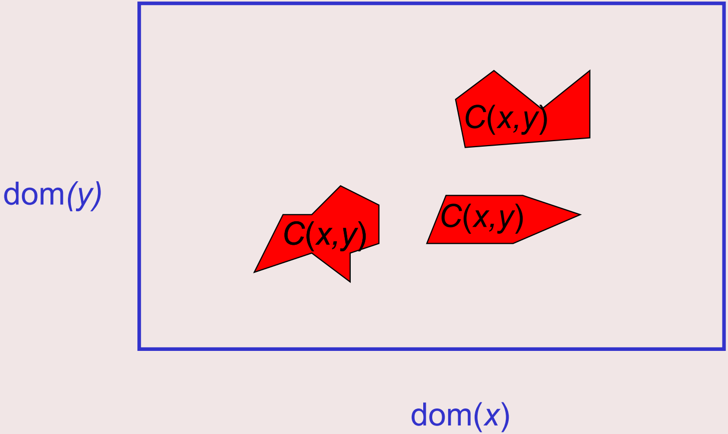
Decomposing constraint `distinct`$([v_1, \ldots, v_q])$ into $\frac{q \cdot (q-1)}{2}$ constraints $v_i \neq v_j$ $(1 \leq i < j \leq q)$ yields VC for `distinct` and requires $\mathcal{O}(q^2)$ space: ☞ see Topic 16: Propagators.

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 has $3 \cdot 3 \cdot 2 - 3$ non-solutions!

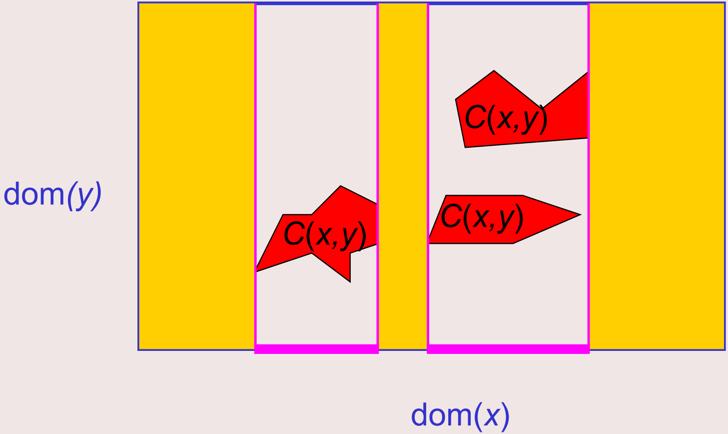
☞ CP = reasoning with sets of (at least all) possible values!

Geometric intuition (pictures: © Yves Deville)



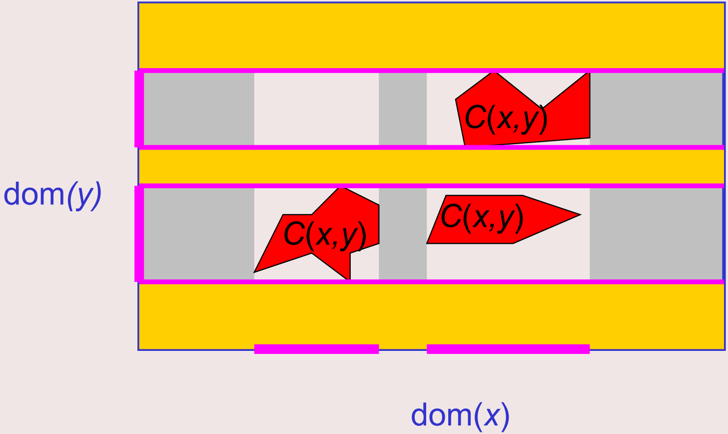
In general, a domain is a union of intervals.

Geometric intuition (pictures: © Yves Deville)



Contracting the domain of x from 1 interval into 2 intervals

Geometric intuition (pictures: © Yves Deville)



Contracting the domain of y from 1 interval into 2 intervals

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 arc consistency (AC) in the case of binary constraints (of arity 2).
- DC is the strongest consistency, and thus implies VC for instance, but enforcing it is sometimes prohibitively expensive, for instance on linear equality constraints.
- Naïve ways to enforce DC for a constraint *c* are:
  - Execute the DC definition as an algorithm with 3 nested loops to find values staying in the variable domains of *c*.
  - Compute all solutions to *c* and lose them by projection onto each variable domain: see example at slide 15.

Both are impractical! ☞ It is often possible to exploit the combinatorial structure of a constraint to enforce DC much faster: examples are in Topic 16: Propagators.

Outline

1. Definitions
2. Value Consistency
3. Domain Consistency
4. **Bounds Consistency**
5. Consistency and Backtracking

Bounds Consistency

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

Consider the store $s = \{x \mapsto \{1, 2, 6\}, \; y \mapsto \{0, 2, 3, 4, 5\}\}$:

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

Definitions

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 inclusive 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.

UPPSALA UNIVERSITET

Definition

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 $\gamma(\cdots)$.

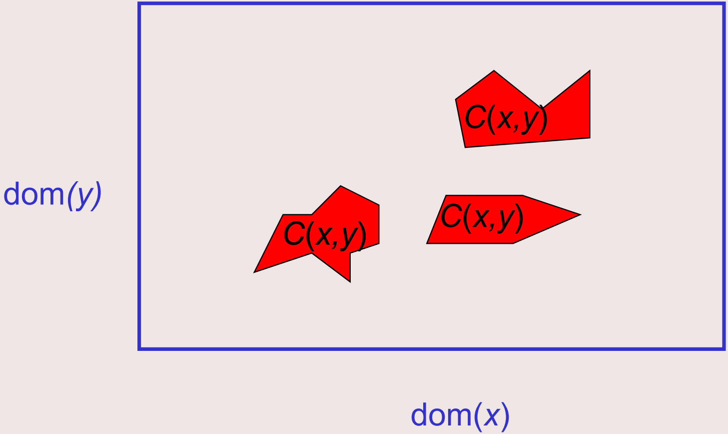
Note that bounds(D) is *not* a misspelling of bounds($\mathbb{D}$).

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 $\{x \mapsto \{2, 3, 5\}\,,\ y, z \mapsto \{4, 6\}\}$.

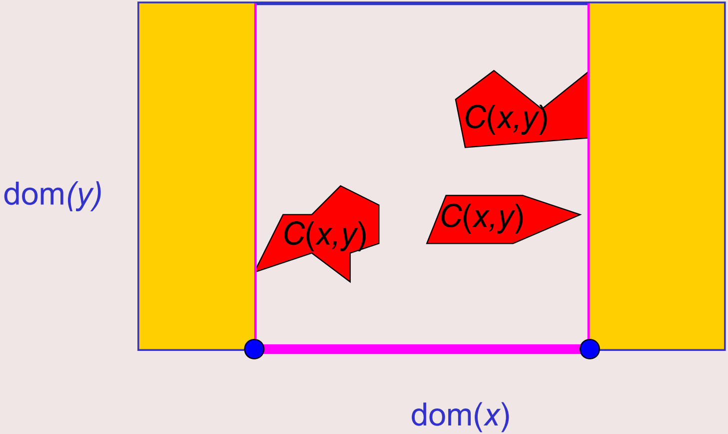
Geometric intuition (pictures: © Yves Deville)



In general, a domain is a union of intervals.

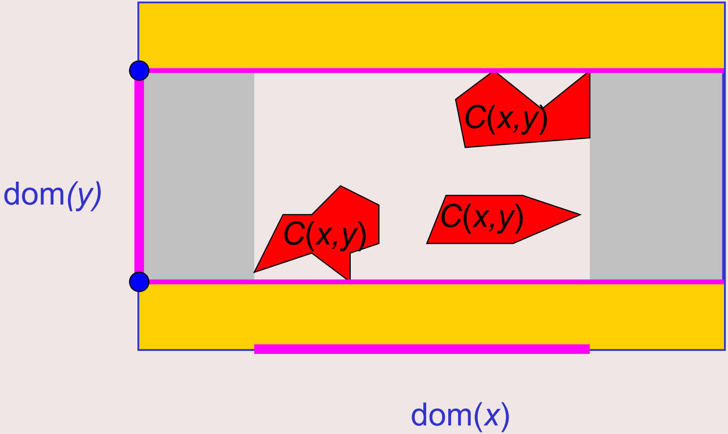
UPPSALA UNIVERSITET

Geometric intuition (pictures: © Yves Deville)



Contracting the domain of x into a tighter interval

Geometric intuition (pictures: © Yves Deville)



Contracting the domain of y into a tighter interval

UPPSALA UNIVERSITET

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($\mathbb{R}$) for arithmetic constraints. Note:

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

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

Example (Consistency for SEND + MORE = MONEY)

Enforcing DC for 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 for `distinct`$(\cdots)$ and BC($\mathbb{R}$) for the linear equality, as enforcing DC for linear equality is prohibitively expensive (see Topic 16: Propagators).

Outline

More about Consistency

Definitions

- The existentially quantified values in the definitions of DC and BC($\cdot$) 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 supported, else d is unsupported.

Definitions

The enforcing of some consistency for some constraint is called propagation and is performed by an algorithm called a propagator, which targets to delete the unsupported domain values according to that consistency.

☞ See in-depth discussion in Topic 14: Propagation.
☞ See example propagators in Topic 16: Propagators.

- *Not* all propagators enforce VC, BC($\cdot$), or DC, which have simple definitions: there are many useful but unnamed consistencies that can be, and are, enforced.

  For example, DC can be enforced for *some* variables of the constraint, and BC($\cdot$) for its *remaining* variables.
- Pragmatism often prevails in propagator design: maximally contract domains within a reasonable time and space complexity, typically *polynomial* in the number of decision variables of the constraint.
- A CP solver may offer *a few* propagators / consistencies for a constraint predicate, one being the default.
- The modeller *must* make experiments to choose for each constraint a suitable propagator / consistency, given typical instances of the problem at hand.
- A variable can be subjected to *several* consistencies in the constraints it participates in: the data structure for stores is *not* specific to a particular consistency.

Complexity of Consistencies

Preview of Topic 16: Propagators:

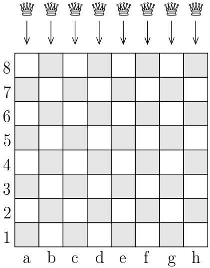
Example ($\texttt{distinct}([v_1, \ldots, v_q])$)

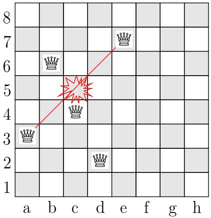
- Value consistency: $\mathcal{O}(q)$ time
- Bounds consistency: $\mathcal{O}(q \cdot \lg q)$ time; often $\mathcal{O}(q)$ time
- Domain consistency: $\mathcal{O}(m \cdot \sqrt{q})$ time, $\mathcal{O}(m \cdot q)$ space, for $m \geq q$ domain values

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 ($<, \leq, \geq, >$)

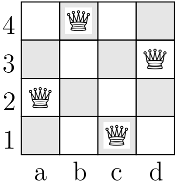
n-Queens Revisited (pics: © Ch. Lecoutre)

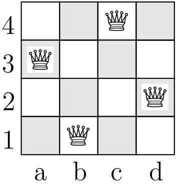




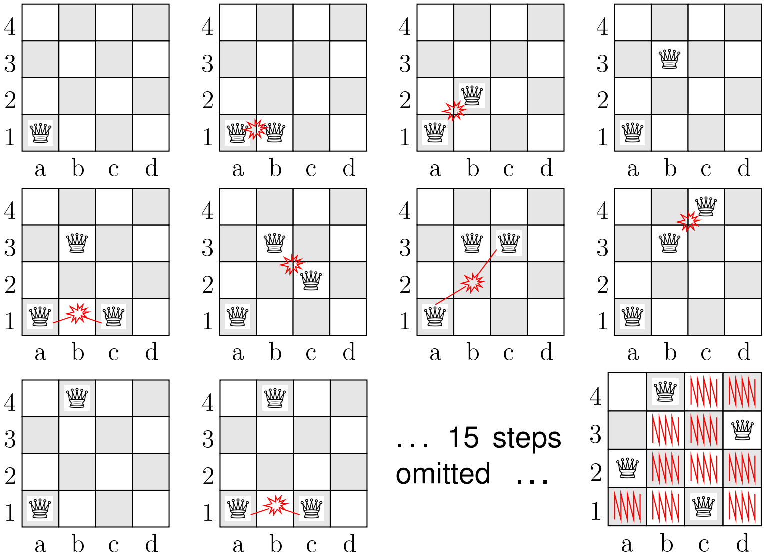
$$\texttt{distinct}([r_a, r_b, .., r_h]), \texttt{distinct}([|r_a-1|, |r_b-2|, \dots, |r_h-8|])$$

The two solutions to the 4-queens instance:





4-Queens: Backtracking Search (BT)



UPPSALA UNIVERSITET

Definitions

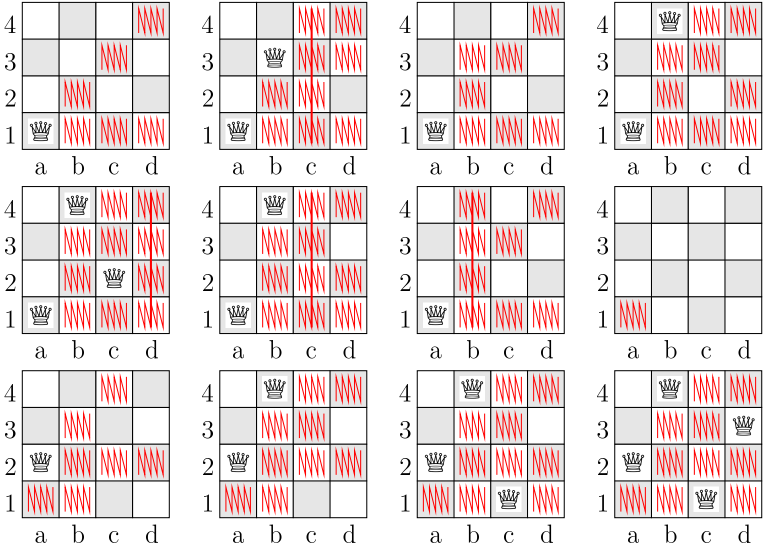
Value Consistency

Domain Consistency

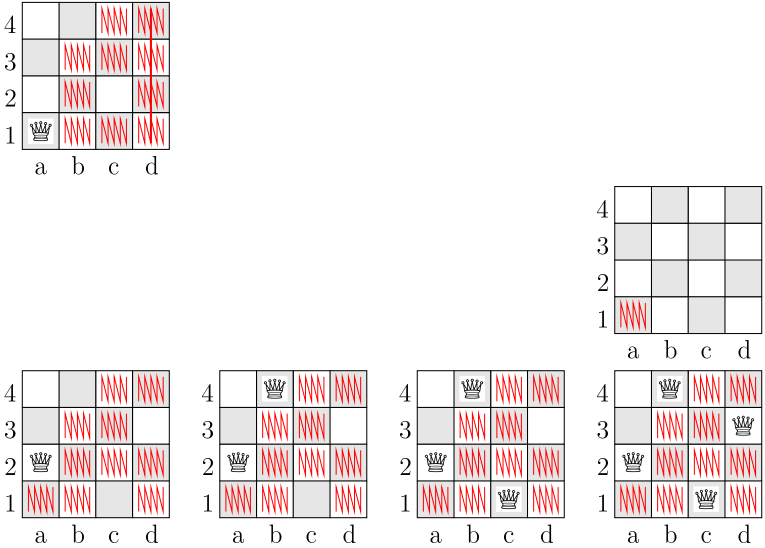
Bounds Consistency

Consistency and Backtracking

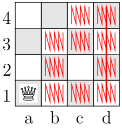
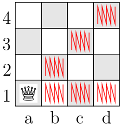
4-Queens: BT + Value Consistency (VC)



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 $\{r_a \mapsto \{1\}\,,\ r_b, r_c, r_d \mapsto \{2,3,4\}\}$, like under VC.
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 fixed variable and the remaining variables, but DC also detects the conflicts between the remaining variables.