Symmetry

Marco Kuhlmann & Guido Tack

Lecture 10

Plan for today

- ROBDDs for finite set variables
- Symmetries in CSPs
- Avoiding symmetry

ROBDDs for finite set constraints

Set domains as Boolean functions

- Characteristic function of a set:

$$\chi_S(i) \Leftrightarrow i \in S$$

- Sets of sets: disjunction of characteristic functions

$$\chi_{\mathscr{S}}(i) \Leftrightarrow \bigvee_{S \in \mathscr{S}} \chi_S(i)$$

Example

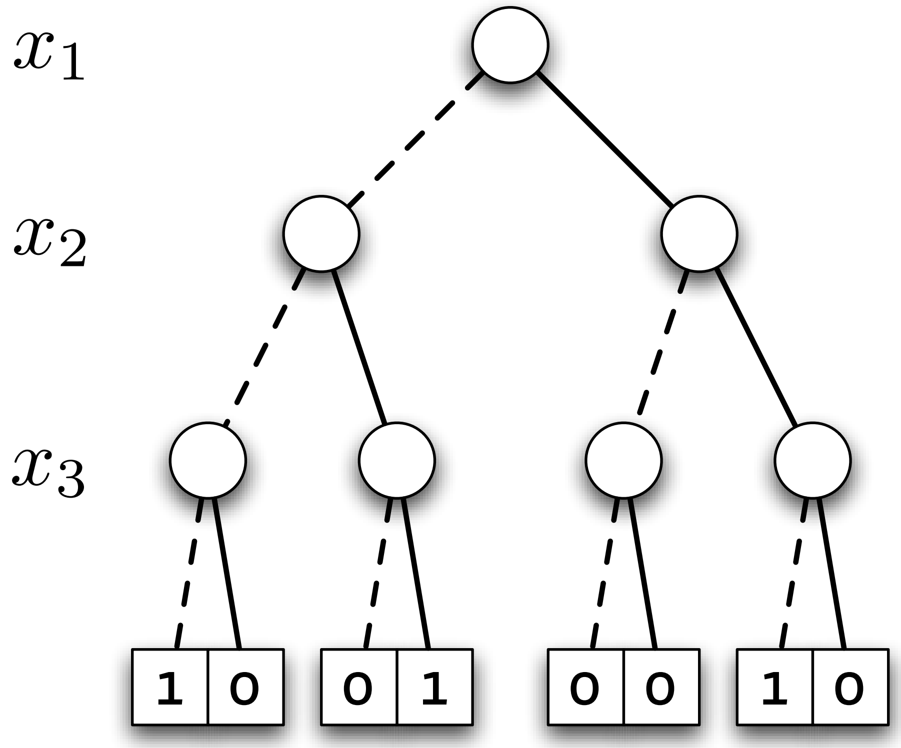
- Consider the domain $\{\{\}, \{1,2\}, \{2,3\}\}$
- Introduce propositional variables x_1, x_2, x_3
- Represent single variable domain as

$$(\neg x_1 \wedge \neg x_2 \wedge \neg x_3) \vee (x_1 \wedge x_2 \wedge \neg x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3)$$

- Represent all variable domains as conjunction
- Efficient datastructure: ROBDDs

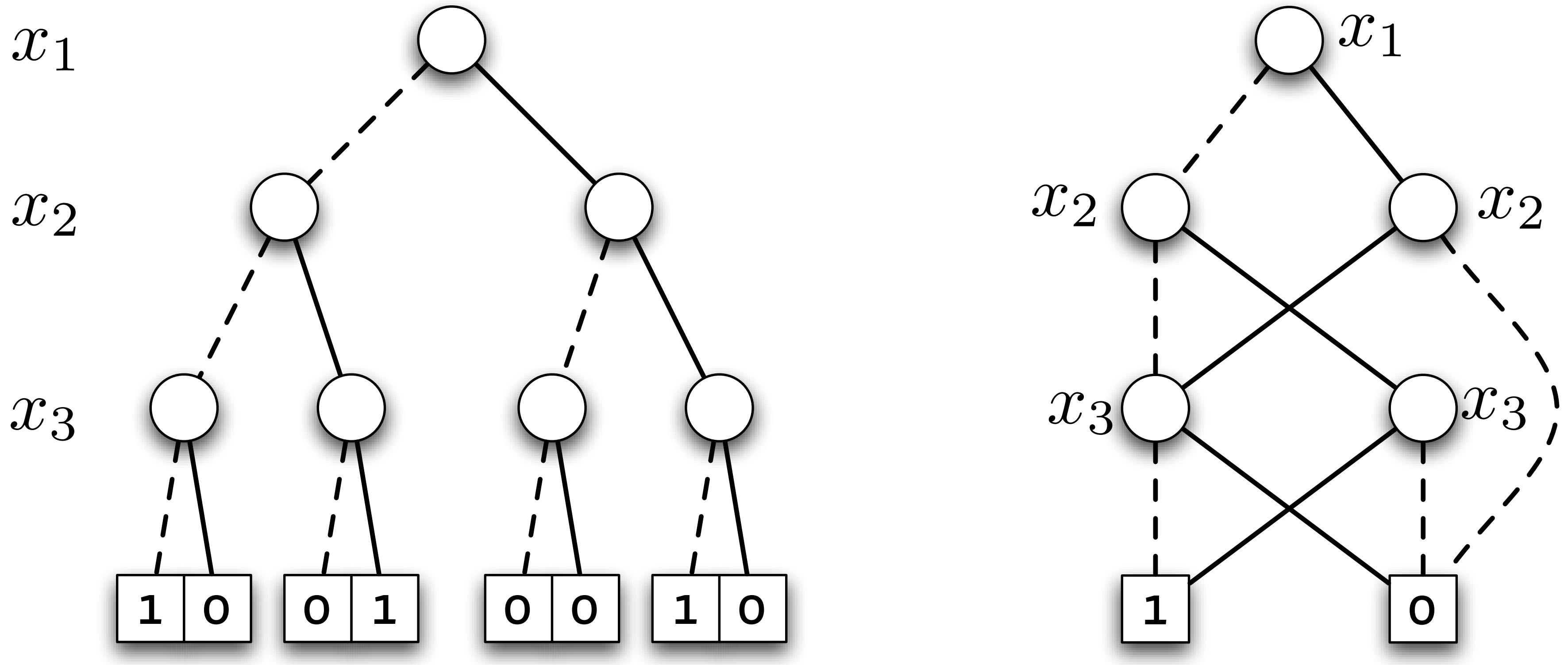
ROBDDs

- Canonical representation for Boolean functions
- Reduced decision tree with fixed variable order



ROBDDs

- Canonical representation for Boolean functions
- Reduced decision tree with fixed variable order



Operations on ROBDDs

- Conjunction, disjunction, implication, ...
- Projection $\exists x.\phi$

Constraints as ROBDDs

- Consider the constraint $X \subseteq Y$
- Written as a formula: $\forall v \in \mathcal{U} : v \in X \Rightarrow v \in Y$
- With fixed universe $\mathcal{U} = \{1, 2, 3\}$:

$$(X_1 \Rightarrow Y_1) \wedge (X_2 \Rightarrow Y_2) \wedge (X_3 \Rightarrow Y_3)$$

- Boolean formula! Represent as ROBDD

Propagation

- For each variable X we have a domain representation χ_X
- Our propagator is represented as φ
- Projection propagator for Y:

$$\exists \mathscr{X} \setminus \{Y\}. \bigwedge_{X \in \mathscr{X}} \chi_X \wedge \varphi$$

Propagation: example

- Subset constraint on U={1,2,3}:

  $(X_1 \Rightarrow Y_1) \wedge (X_2 \Rightarrow Y_2) \wedge (X_3 \Rightarrow Y_3)$

- Current domain:

  $\{X \mapsto \{\emptyset, \{1,2\}, \{2,3\}\}, Y \mapsto \{\{2,3\}, \{3\}\}\}$

$$D = ((\overline{X_1 X_2 X_3}) \vee (X_1 X_2 \overline{X_3}) \vee (\overline{X_1} X_2 X_3)) \wedge ((\overline{Y_1} Y_2 Y_3) \vee (\overline{Y_1 Y_2} Y_3))$$

- Propagation:

  $\exists Y_1, Y_2, Y_3.\ D \wedge (X_1 \Rightarrow Y_1) \wedge (X_2 \Rightarrow Y_2) \wedge (X_3 \Rightarrow Y_3)$

  $\exists X_1, X_2, X_3.\ D \wedge (X_1 \Rightarrow Y_1) \wedge (X_2 \Rightarrow Y_2) \wedge (X_3 \Rightarrow Y_3)$

Pros and Cons

- Propagation is complete
- Propagators are compositional:
  - conjunction and disjunction of constraints easily expressible
- Possibly expensive
  - ROBDD operations worst case exponential
- Some constraints have exp. size formulas

Literature

- Hawkins, Lagoon, Stuckey. *Solving Set Constraint Satisfaction Problems using ROBDDs.* JAIR Volume 24, 2005
- Bryant. *Symbolic Boolean manipulation with ordered binary-decision diagrams.* ACM Comput. Surv., 24 (3), 1992

Symmetries

Social golfers

- **Problem:**

  w weeks, g groups of p players each

  all players play once a week

  no two players in the same group more than once

- **Model:**

  $w \times g \times p$ integer variables

Example: Social golfers

	group 1			group 2			group 3			group 4			group 5		
week 1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
week 2	0	3	6	1	4	9	2	7	12	5	10	13	8	11	14
week 3	0	4	13	1	3	11	2	6	10	5	8	12	7	9	14
week 4	0	5	14	1	10	12	2	3	8	4	7	11	6	9	13
week 5	0	7	10	1	8	13	2	4	14	3	9	12	5	6	11
week 6	0	8	9	1	5	7	2	11	13	3	10	14	4	6	12
week 7	0	11	12	1	6	14	2	5	9	3	7	13	4	8	10

Example: Social golfers

	group 1			group 2			group 3			group 4			group 5		
week 1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
week 2	0	3	6	1	4	9	2	7	12	5	10	13	8	11	14
week 3	0	4	13	1	3	11	2	6	10	5	8	12	7	9	14
week 4	0	5	14	1	10	12	2	3	8	4	7	11	6	9	13
week 5	0	7	10	1	8	13	2	4	14	3	9	12	5	6	11
week 6	0	8	9	1	5	7	2	11	13	3	10	14	4	6	12
week 7	0	11	12	1	6	14	2	5	9	3	7	13	4	8	10

Example: Social golfers

	group 1			group 2			group 3			group 4			group 5		
week 1	2	1	0	3	4	5	6	7	8	9	10	11	12	13	14
week 2	6	3	0	1	4	9	2	7	12	5	10	13	8	11	14
week 3	13	4	0	1	3	11	2	6	10	5	8	12	7	9	14
week 4	14	5	0	1	10	12	2	3	8	4	7	11	6	9	13
week 5	10	7	0	1	8	13	2	4	14	3	9	12	5	6	11
week 6	9	8	0	1	5	7	2	11	13	3	10	14	4	6	12
week 7	12	11	0	1	6	14	2	5	9	3	7	13	4	8	10

Example: Social golfers

week 1	2	1	0	3	4	5	6	7	8	9	10	11	12	13	14
week 2	6	3	0	1	4	9	2	7	12	5	10	13	8	11	14
week 3	13	4	0	1	3	11	2	6	10	5	8	12	7	9	14
week 4	14	5	0	1	10	12	2	3	8	4	7	11	6	9	13
week 5	10	7	0	1	8	13	2	4	14	3	9	12	5	6	11
week 6	9	8	0	1	5	7	2	11	13	3	10	14	4	6	12
week 7	12	11	0	1	6	14	2	5	9	3	7	13	4	8	10
	group 1			group 2			group 3			group 4			group 5		

Example: Social golfers

	group 1			group 2			group 3			group 4			group 5		
week 1	2	1	0	3	4	5	6	7	8	9	10	11	12	13	14
week 2	6	3	0	1	4	9	2	7	12	5	10	13	8	11	14
week 3	13	4	0	1	3	11	2	6	10	5	8	12	7	9	14
week 4	14	5	0	1	10	12	2	3	8	4	7	11	6	9	13
week 5	10	7	0	1	8	13	2	4	14	3	9	12	5	6	11
week 6	9	8	0	1	5	7	2	11	13	3	10	14	4	6	12
week 7	12	11	0	1	6	14	2	5	9	3	7	13	4	8	10

Example: Social golfers

	group 1			group 2			group 3			group 4			group 5		
week 1	2	1	0	9	10	11	6	7	8	3	4	5	12	13	14
week 2	6	3	0	5	10	13	2	7	12	1	4	9	8	11	14
week 3	13	4	0	5	8	12	2	6	10	1	3	11	7	9	14
week 4	14	5	0	4	7	11	2	3	8	1	10	12	6	9	13
week 5	10	7	0	3	9	12	2	4	14	1	8	13	5	6	11
week 6	9	8	0	3	10	14	2	11	13	1	5	7	4	6	12
week 7	12	11	0	3	7	13	2	5	9	1	6	14	4	8	10

Example: Social golfers

	group 1			group 2			group 3			group 4			group 5		
week 1	2	1	0	9	10	11	6	7	8	3	4	5	12	13	14
week 2	6	3	0	5	10	13	2	7	12	1	4	9	8	11	14
week 3	13	4	0	5	8	12	2	6	10	1	3	11	7	9	14
week 4	14	5	0	4	7	11	2	3	8	1	10	12	6	9	13
week 5	10	7	0	3	9	12	2	4	14	1	8	13	5	6	11
week 6	9	8	0	3	10	14	2	11	13	1	5	7	4	6	12
week 7	12	11	0	3	7	13	2	5	9	1	6	14	4	8	10

Example: Social golfers

	group 1			group 2			group 3			group 4			group 5		
week 1	2	1	0	9	10	11	6	7	8	3	4	5	12	13	14
week 2	6	3	0	5	10	13	2	7	12	1	4	9	8	11	14
week 3	13	4	0	5	8	12	2	6	10	1	3	11	7	9	14
week 4	14	5	0	4	7	11	2	3	8	1	10	12	6	9	13
week 5	10	7	0	3	9	12	2	4	14	1	8	13	5	6	11
week 6	9	8	0	3	10	14	2	11	13	1	5	7	4	6	12
week 7	12	11	0	3	7	13	2	5	9	1	6	14	4	8	10

Example: Social golfers

	group 1			group 2			group 3			group 4			group 5		
week 1	2	1	0	9	10	11	6	7	8	3	4	5	12	13	14
week 2	10	7	0	3	9	12	2	4	14	1	8	13	5	6	11
week 3	13	4	0	5	8	12	2	6	10	1	3	11	7	9	14
week 4	14	5	0	4	7	11	2	3	8	1	10	12	6	9	13
week 5	6	3	0	5	10	13	2	7	12	1	4	9	8	11	14
week 6	9	8	0	3	10	14	2	11	13	1	5	7	4	6	12
week 7	12	11	0	3	7	13	2	5	9	1	6	14	4	8	10

Example: Social golfers

	group 1			group 2			group 3			group 4			group 5		
week 1	2	1	0	9	10	11	6	7	8	3	4	5	12	13	14
week 2	10	7	0	3	9	12	2	4	14	1	8	13	5	6	11
week 3	13	4	0	5	8	12	2	6	10	1	3	11	7	9	14
week 4	14	5	0	4	7	11	2	3	8	1	10	12	6	9	13
week 5	6	3	0	5	10	13	2	7	12	1	4	9	8	11	14
week 6	9	8	0	3	10	14	2	11	13	1	5	7	4	6	12
week 7	12	11	0	3	7	13	2	5	9	1	6	14	4	8	10

Example: Social golfers

	group 1			group 2			group 3			group 4			group 5		
week 1	2	1	0	9	10	11	6	7	8	3	4	5	12	13	14
week 2	10	7	0	3	9	12	2	4	14	1	8	13	5	6	11
week 3	13	4	0	5	8	12	2	6	10	1	3	11	7	9	14
week 4	14	5	0	4	7	11	2	3	8	1	10	12	6	9	13
week 5	6	3	0	5	10	13	2	7	12	1	4	9	8	11	14
week 6	9	8	0	3	10	14	2	11	13	1	5	7	4	6	12
week 7	12	11	0	3	7	13	2	5	9	1	6	14	4	8	10

Example: Social golfers

	group 1			group 2			group 3			group 4			group 5		
week 1	2	1	0	7	10	11	6	9	8	3	4	5	12	13	14
week 2	10	9	0	3	7	12	2	4	14	1	8	13	5	6	11
week 3	13	4	0	5	8	12	2	6	10	1	3	11	9	7	14
week 4	14	5	0	4	9	11	2	3	8	1	10	12	6	7	13
week 5	6	3	0	5	10	13	2	9	12	1	4	7	8	11	14
week 6	7	8	0	3	10	14	2	11	13	1	5	9	4	6	12
week 7	12	11	0	3	9	13	2	5	7	1	6	14	4	8	10

Example: Social golfers

week 1	2	1	0	7	10	11	6	9	8	3	4	5	12	13	14
week 2	10	9	0	3	7	12	2	4	14	1	8	13	5	6	11
week 3	13	4	0	5	8	12	2	6	10	1	3	11	9	7	14
week 4	14	5	0	4	9	11	2	3	8	1	10	12	6	7	13
week 5	6	3	0	5	10	13	2	9	12	1	4	7	8	11	14
week 6	7	8	0	3	10	14	2	11	13	1	5	9	4	6	12
week 7	12	11	0	3	9	13	2	5	7	1	6	14	4	8	10
	group 1			group 2			group 3			group 4			group 5		

Example: Social golfers

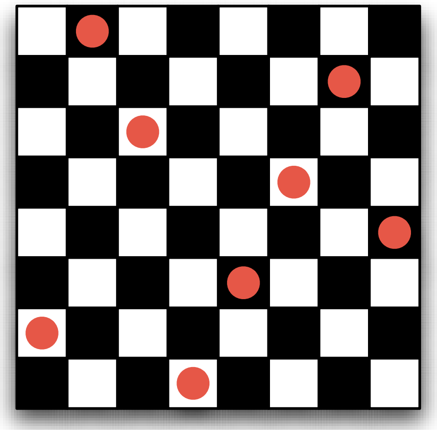
2	1	0	7	10	11	6	9	8	3	4	5	12	13	14
10	9	0	3	7	12	2	4	14	1	8	13	5	6	11
13	4	0	5	8	12	2	6	10	1	3	11	9	7	14
14	5	0	4	9	11	2	3	8	1	10	12	6	7	13
6	3	0	5	10	13	2	9	12	1	4	7	8	11	14
7	8	0	3	10	14	2	11	13	1	5	9	4	6	12
12	11	0	3	9	13	2	5	7	1	6	14	4	8	10

permuting weeks and groups: 7! × 5! = 604.800

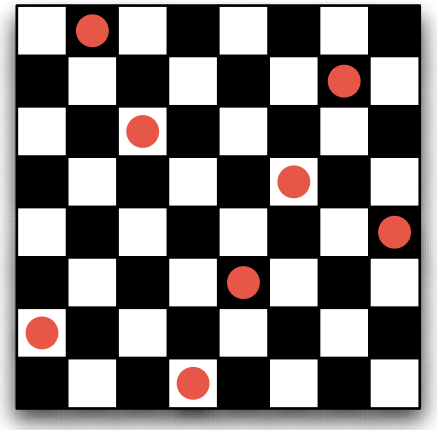
just permuting all players: 15! = 1.307.674.368.000

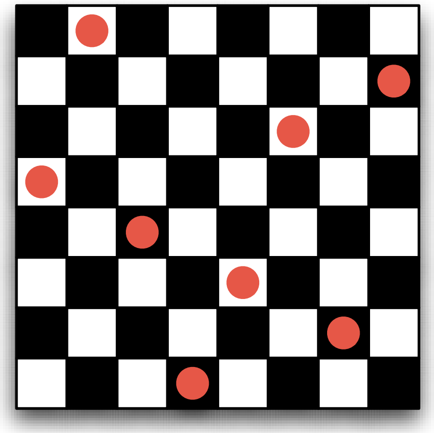
Example: Queens

id

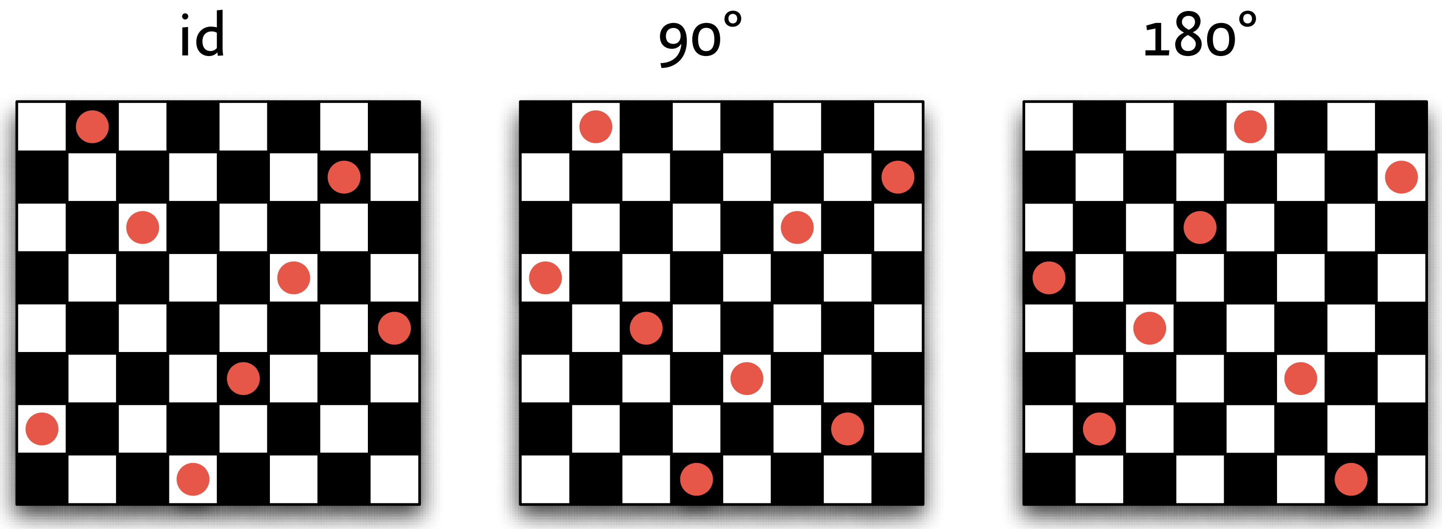


Example: Queens



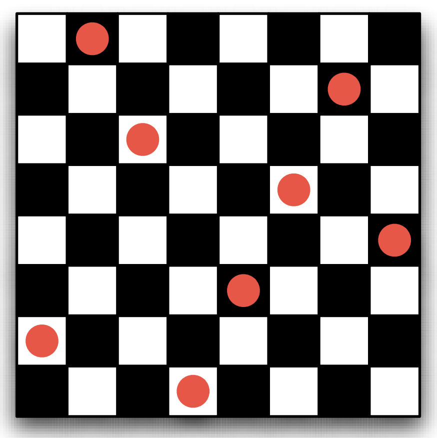


Example: Queens

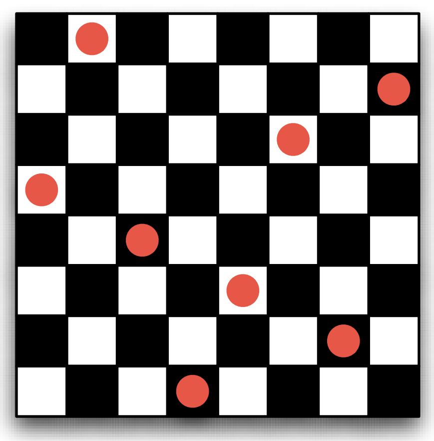


Example: Queens

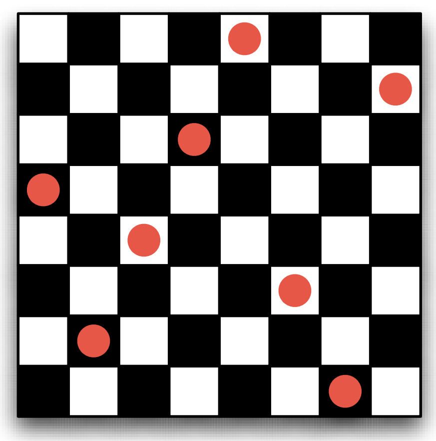
id



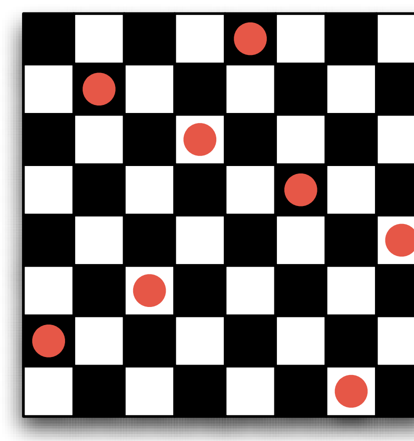
90°



180°

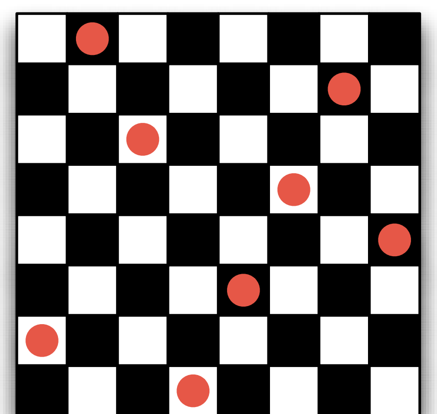


270°

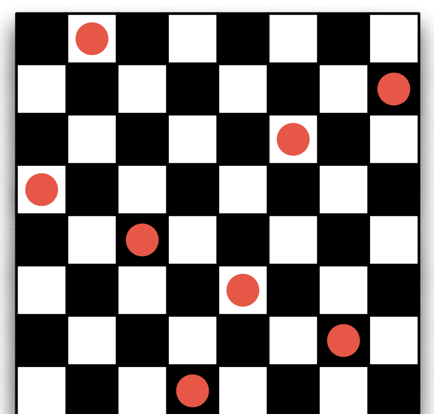


Example: Queens

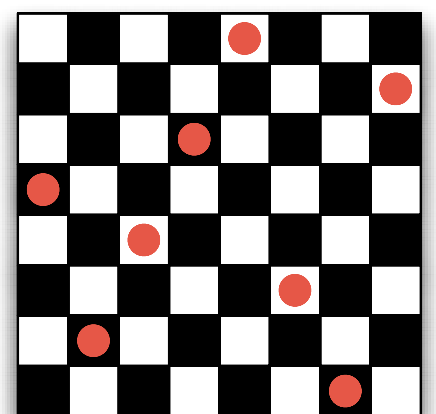
id



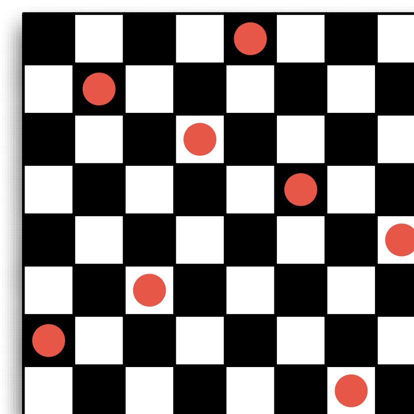
90°

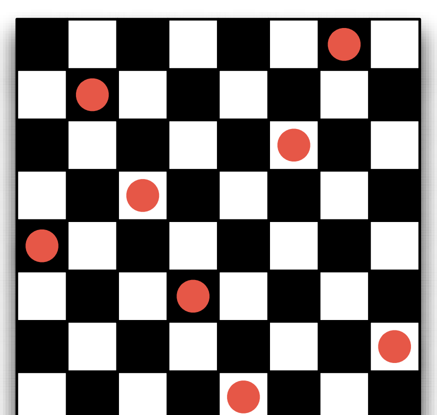


180°



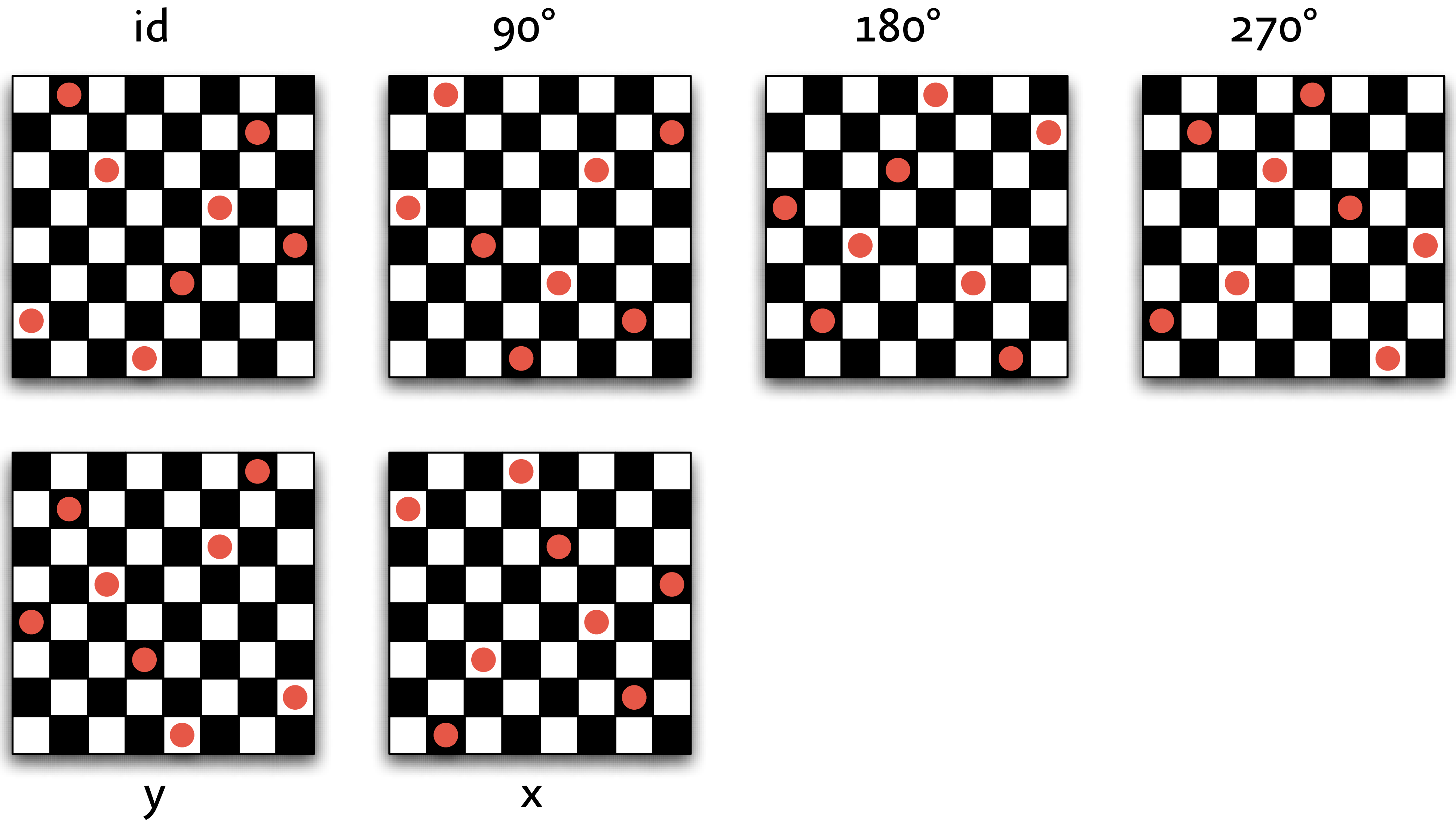
270°



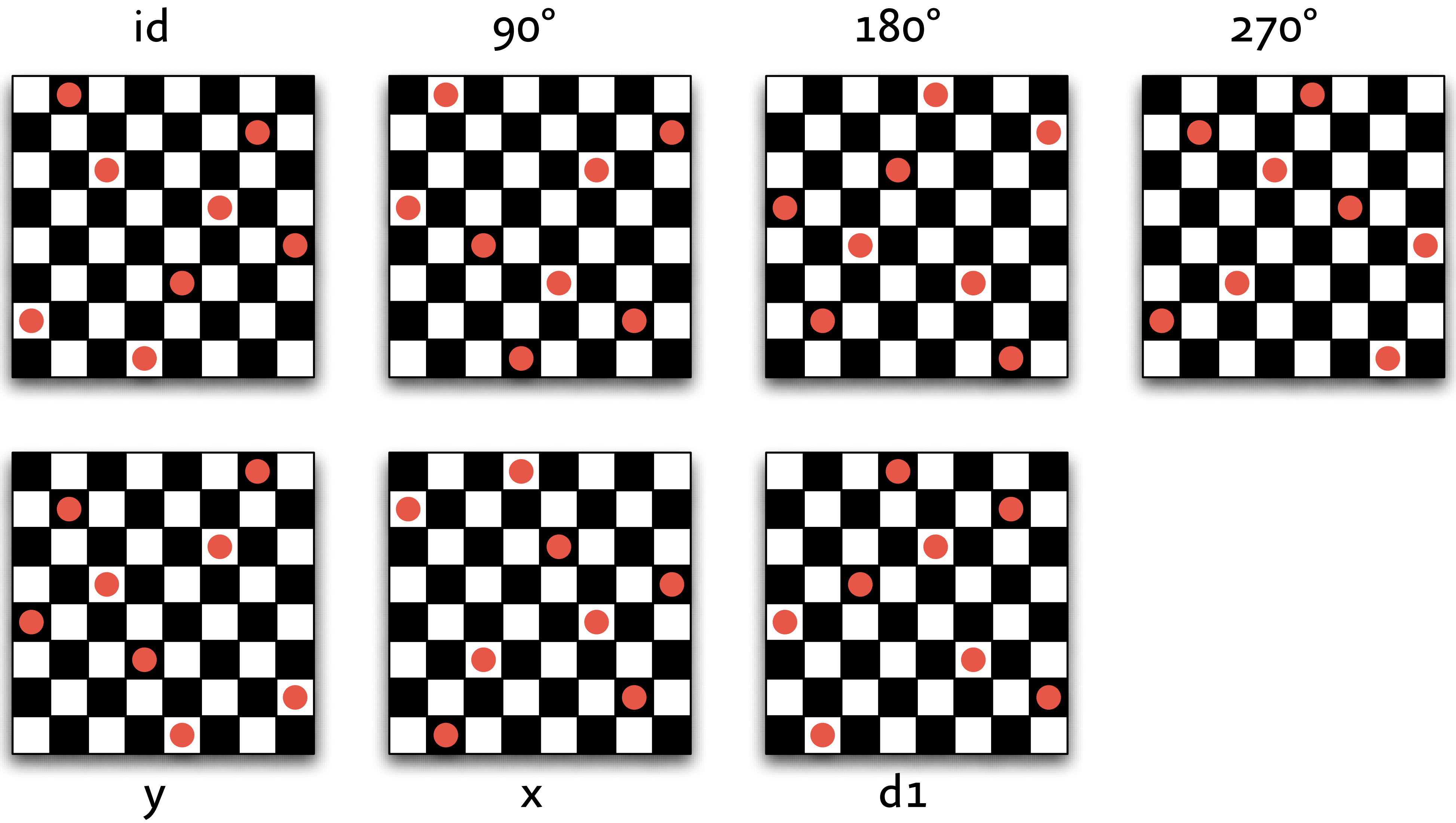


y

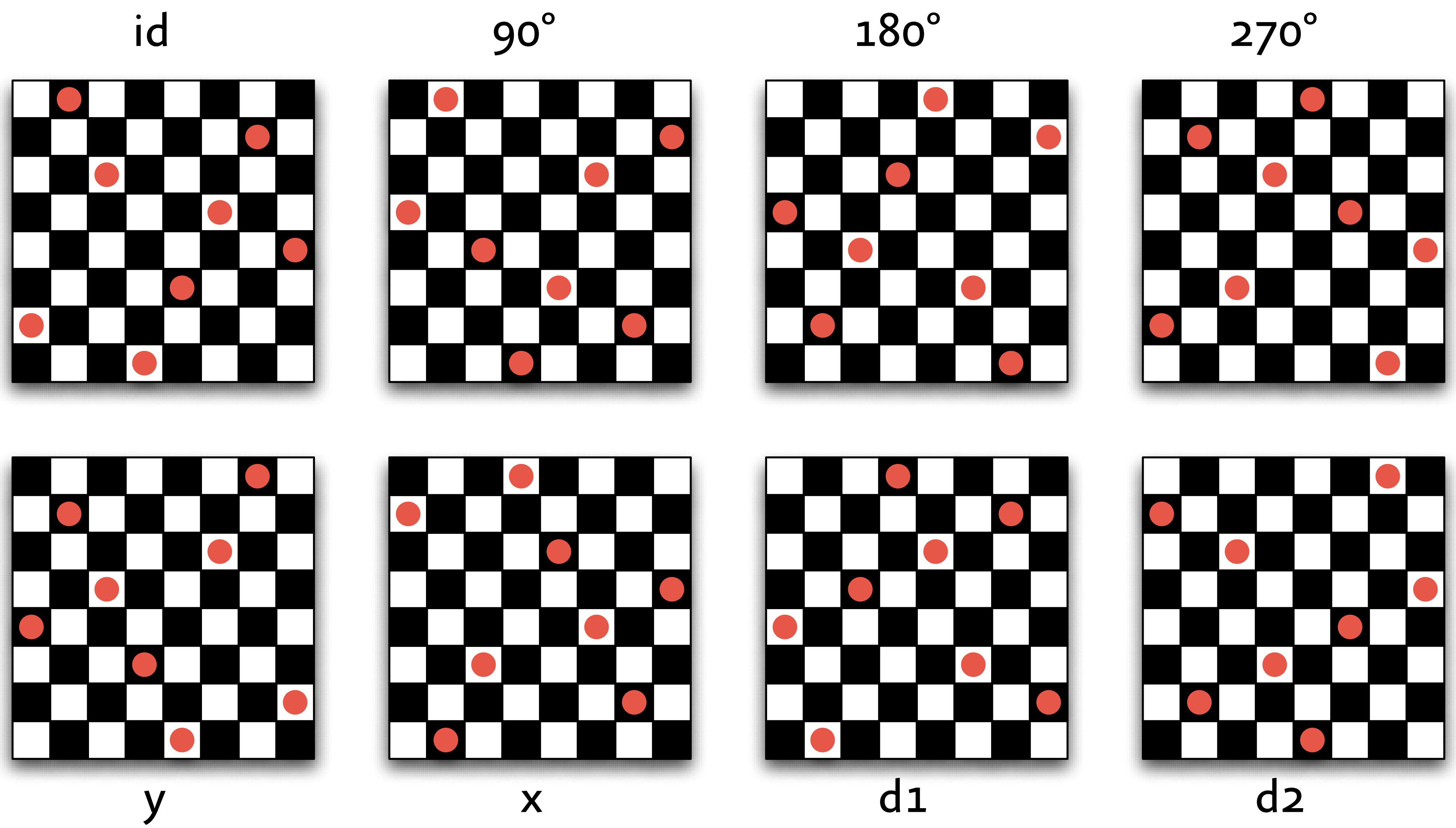
Example: Queens



Example: Queens

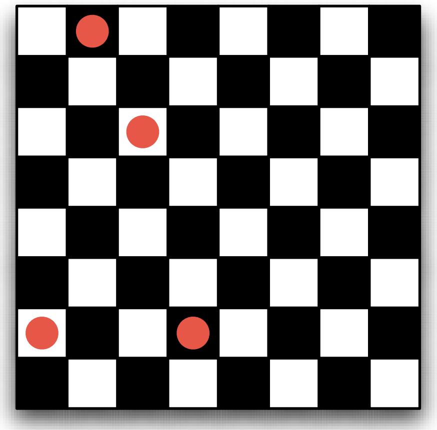


Example: Queens

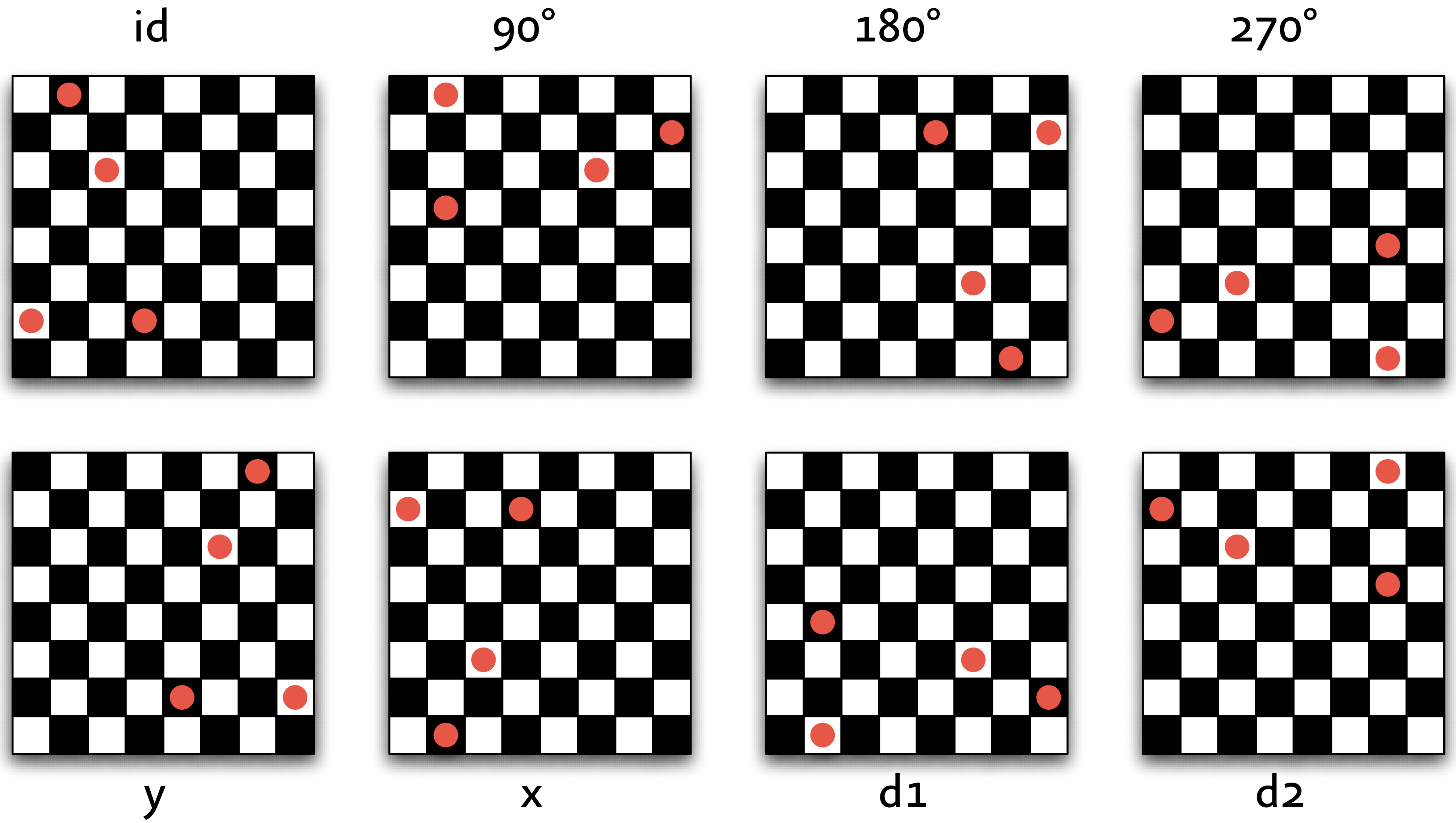


Symmetric failure

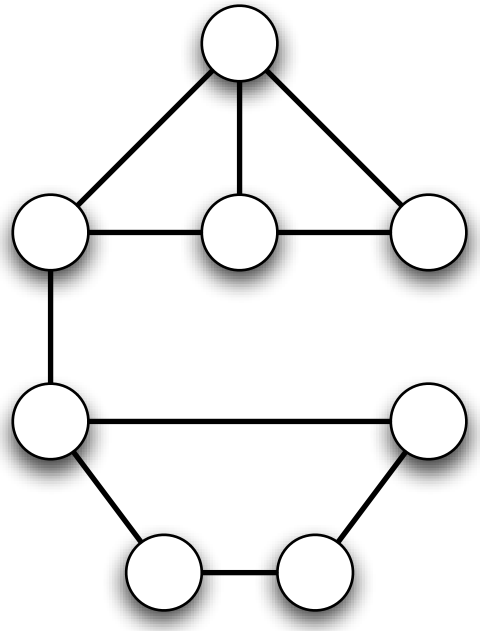
id



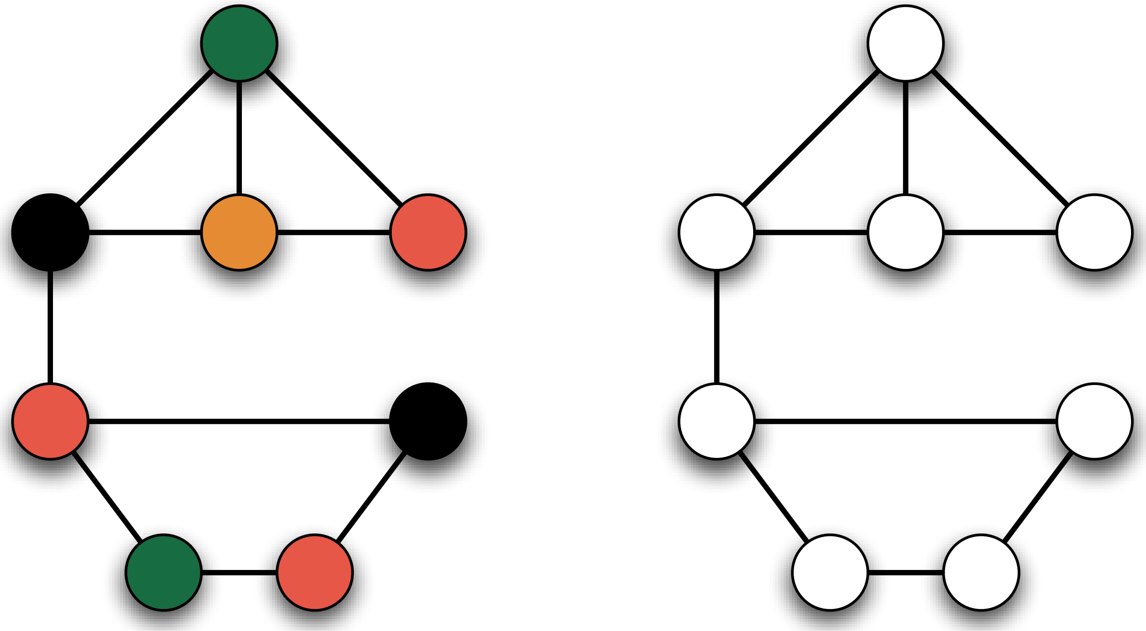
Symmetric failure



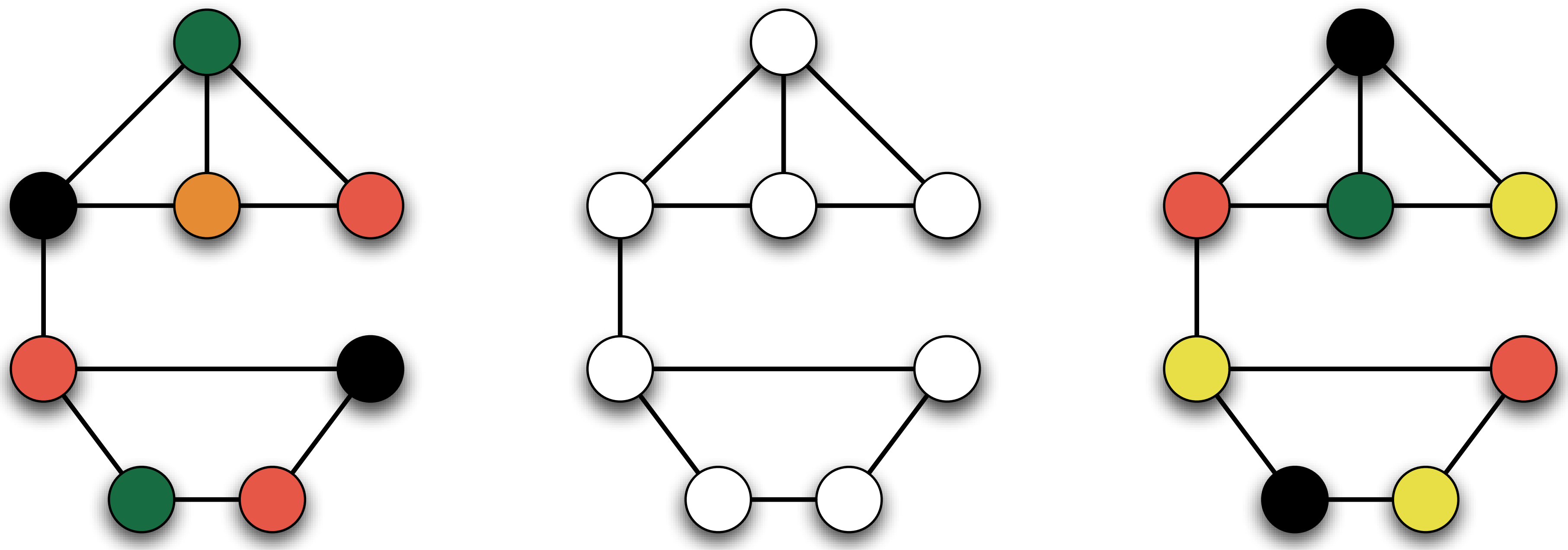
Example: graph colouring



Example: graph colouring



Example: graph colouring



Kinds of symmetries

- **Variable symmetry:**

  permuting variables keeps solutions invariant

  $\{x_i \mapsto v_i\} \in \mathrm{sol}(P) \Leftrightarrow \{x_{\sigma(i)} \mapsto v_i\} \in \mathrm{sol}(P)$

- **Value symmetry:**

  permuting values keeps solutions invariant

  $\{x_i \mapsto v_i\} \in \mathrm{sol}(P) \Leftrightarrow \{x_i \mapsto \sigma(v_i)\} \in \mathrm{sol}(P)$

Kinds of symmetries

- **Variable/value symmetry:**

  permute both variables and values

  $\{x_i \mapsto v_i\} \in \mathrm{sol}(P) \Leftrightarrow \{x_{\sigma(i)} \mapsto \sigma'(v_i)\} \in \mathrm{sol}(P)$

Symmetry

- **Ubiquitous!**
- **Inherent in the problem** (chess board)
- **Artefact of the model** (order of players in a group)
- **Different kinds:**
  - variable symmetry (swapping (sets of) variables)
  - value symmetry (permuting values)

Symmetry

- **How can we avoid it?**
  - ... by model reformulation
  - ... by adding constraints to the model
  - ... during search
  - ... by dominance detection

Avoiding symmetry by reformulation

Use set variables

- **Sets are unordered**
- **Golfers example:** represent groups as sets
- **New model contains no symmetry in groups**
- **… but still a lot of symmetries**

Solve different problem

- Recast your problem into a different problem without symmetries
- Example: all-interval series
  - find permutation of 0...n such that differences between adjacent numbers are a permutation of 1...n

0	10	1	9	2	8	3	7	4	6	5
	10	9	8	7	6	5	4	3	2	1

Solve different problem

- Recast your problem into a different problem without symmetries
- Example: all-interval series
  - find permutation of 0...n such that differences between adjacent numbers are a permutation of 1...n

0	10	1	9	2	8	3	7	4	6	5
	10	9	8	7	6	5	4	3	2	1

Solve different problem

- Recast your problem into a different problem without symmetries
- Example: all-interval series
    - find permutation of 0...n such that differences between adjacent numbers are a permutation of 1...n

0 10 1 9 2 8 | 3 7 4 6 5
10 9 8 7 6 | 5 4 3 2 1

3 7 4 6 5 0 10 1 9 2 8
4 3 2 1 5 10 9 8 7 6

Solve different problem

- **New problem:**

  find permutation of 0...n such that

  - the permutation starts with 0,n,1
  - adjacent differences including x_n-x_0 contain all 1...n, and exactly one difference occurs twice

- **Extract solutions of the original problem**

Solve dual problem

- Let's say we know how to avoid variable symmetries
- But our problem has value symmetries
- Example: graph colouring
- Consider the *dual problem*:
  - for each value introduce a set such that

$$i \in X_v \Leftrightarrow y_i = v$$

(where y_i are the original variables)

Static symmetry breaking

Symmetry breaking constraints

- **Idea:**
  - "break" symmetry by ruling out symmetric solutions
  - add constraints to the original model

Lex-leader constraints

- **Assumption:** domains are ordered
- Let Σ be the set of all variable symmetry permutations
- All **variable symmetry** can be broken by

$$\bigwedge_{\sigma \in \Sigma} [x_1, \ldots, x_n] \leq_{\text{lex}} [x_{\sigma(1)}, \ldots, x_{\sigma(n)}]$$

- Keep only **lexicographically smallest solution**
- Called **lex-leader**

Examples

- **Distinct integers,** $\sigma(1) \neq 1$**:**

  $$[x_1, \ldots, x_n] \leq_{\text{lex}} [x_{\sigma(1)}, \ldots, x_{\sigma(n)}] \Leftrightarrow x_1 < x_{\sigma(1)}$$

- **Disjoint integer sets,** $\sigma(1) \neq 1$**:**

  $$[x_1, \ldots, x_n] \leq_{\text{lex}} [x_{\sigma(1)}, \ldots, x_{\sigma(n)}] \Leftrightarrow \min(x_1) < \min(x_{\sigma(1)})$$

- **Arbitrary integers or sets:** special propagators

  Gecode/J: decompose into rel(this, xs, IRT_LQ, ys)

Examples

- **Queens:**

  $q[0] < q[n-1]$

- **Golfers:**

  $\min(group(w,g)) < \min(group(w,g+1))$

- **All-Interval:**

  $|x[1]-x[0]| > |x[n-1]-x[n-2]|$

What about value symmetries?

- **Same idea:**

$$\bigwedge_{\sigma \in \Sigma} [x_1, \ldots, x_n] \leq_{\text{lex}} [\sigma(x_1), \ldots, \sigma(x_n)]$$

- How implement $\sigma(x_i)$?
- Element constraint!

Example: all-interval series

- $\sigma(v) = n - v$

3 7 4 6 5 0 10 1 9 2 8
4 3 2 1 5 10 9 8 7 6

special case

Example: all-interval series

- $\sigma(v) = n - v$

3 7 4 6 5 0 10 1 9 2 8
4 3 2 1 5 10 9 8 7 6

7 3 6 4 5 10 0 9 1 8 2
4 3 2 1 5 10 9 8 7 6

special case

Example: all-interval series

- $\sigma(v) = n - v$

3 7 4 6 5 0 10 1 9 2 8	7 3 6 4 5 10 0 9 1 8 2
4 3 2 1 5 10 9 8 7 6	4 3 2 1 5 10 9 8 7 6

- $\sigma = [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]$

  $[x_0, \ldots, x_n] \leq_{\text{lex}} [\sigma[x_0], \ldots, \sigma[x_n]]$

special case

Example: all-interval series

- $\sigma(v) = n - v$

3 7 4 6 5 0 10 1 9 2 8	7 3 6 4 5 10 0 9 1 8 2
4 3 2 1 5 10 9 8 7 6	4 3 2 1 5 10 9 8 7 6

- $\sigma = [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]$

  $[x_0, \ldots, x_n] \leq_{\text{lex}} [\sigma[x_0], \ldots, \sigma[x_n]] \quad \Leftrightarrow x_0 < \sigma[x_0]$

special case

Example: all-interval series

- $\sigma(v) = n - v$

3 7 4 6 5 0 10 1 9 2 8
4 3 2 1 5 10 9 8 7 6

7 3 6 4 5 10 0 9 1 8 2
4 3 2 1 5 10 9 8 7 6

- $\sigma = [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]$

$$[x_0, \ldots, x_n] \leq_{\text{lex}} [\sigma[x_0], \ldots, \sigma[x_n]] \quad \Leftrightarrow x_0 < \sigma[x_0]$$

$$\Leftrightarrow x_0 < x_1$$

special case

n-Queens

- $\sigma(v) = n - v$
- $[q_0, \ldots, q_{n-1}] \leq_{\text{lex}} [\sigma[q_0], \ldots, \sigma[1_{n-1}]] \Leftrightarrow q_0 < \sigma[q_0]$

- We have to invest more to break variable/value symmetries

Pros and Cons

- **Good:** for each symmetry, only one solutions remains
- **Bad:**
  - may have to add many constraints
  - remaining solution may not be the first one according to branching heuristic!
- **Especially bad** with dynamic variable selection (like first-fail heuristics!)

SBDS

(Symmetry Breaking During Search)

Adding constraints dynamically

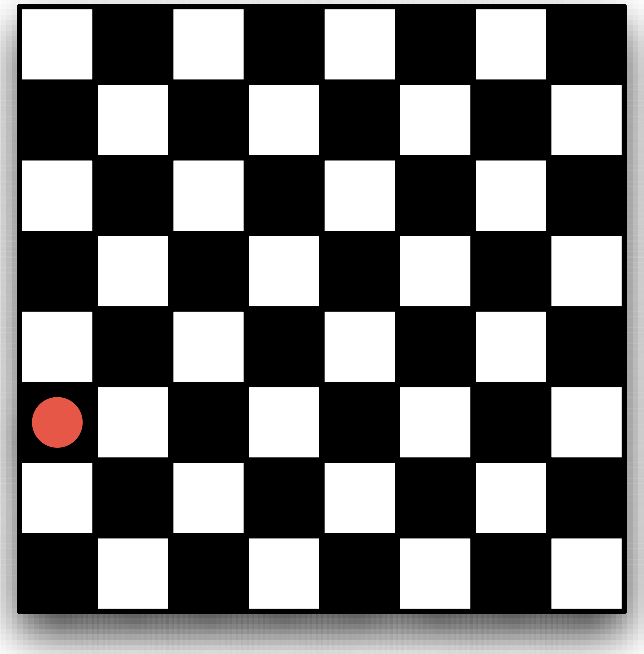
- **Idea:**

  upon backtracking, add constraints that prevent symmetric search states to be visited

- Similar to branch-and-bound search!

- Works for all kinds of symmetries

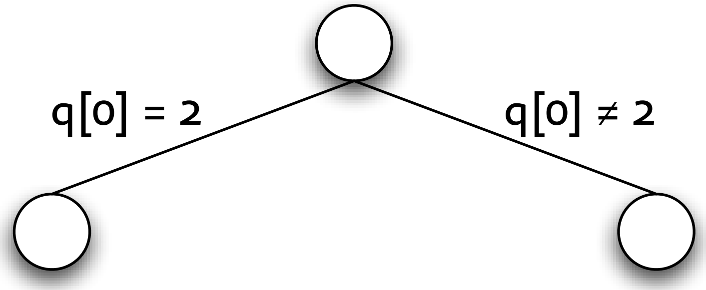
Example: Queens with SBDS

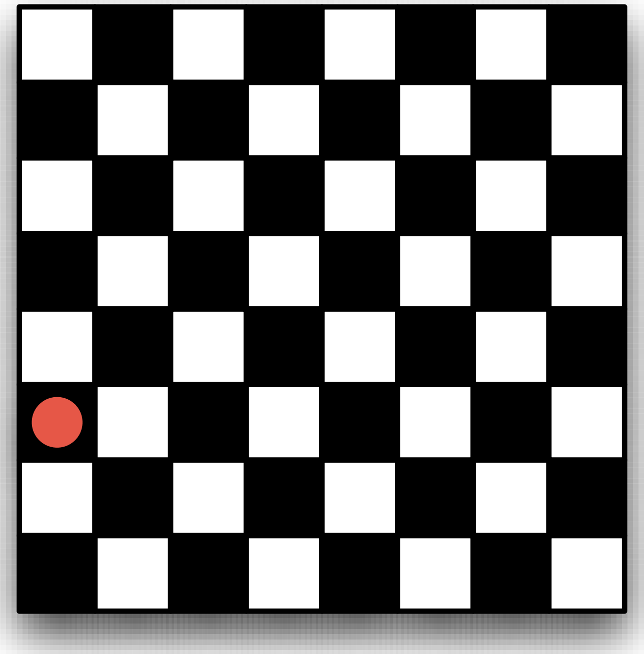


Goal: eliminate r90

$$\{q_i \mapsto j\} \in \text{sol}(\text{queens}) \Leftrightarrow \{q_j \mapsto n - i\} \in \text{sol}(\text{queens})$$

Example: Queens with SBDS

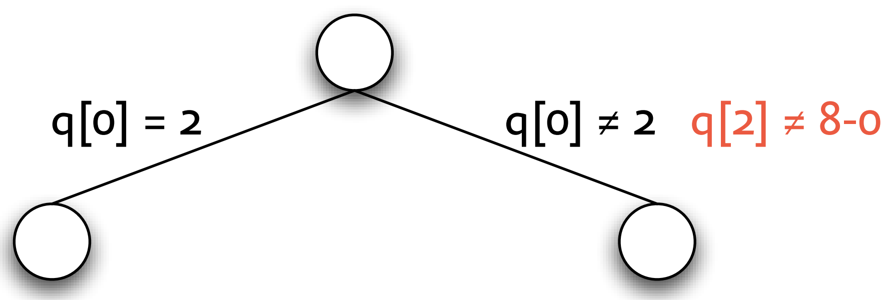


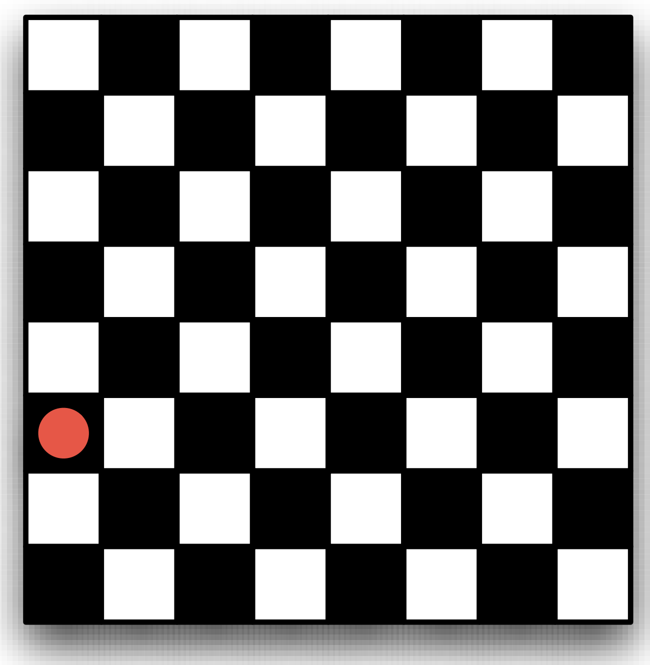


Goal: eliminate r90

$$\{q_i \mapsto j\} \in \mathrm{sol}(\mathrm{queens}) \Leftrightarrow \{q_j \mapsto n - i\} \in \mathrm{sol}(\mathrm{queens})$$

Example: Queens with SBDS

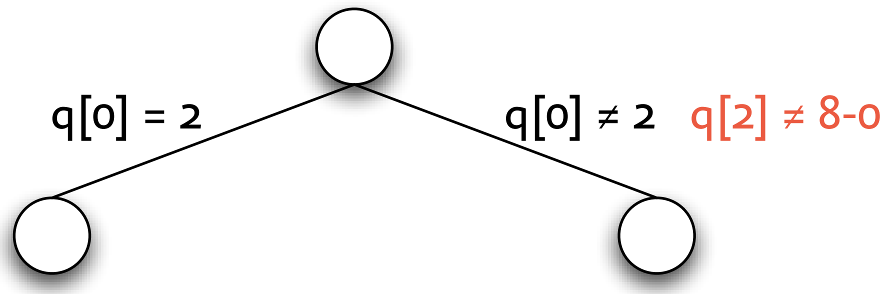


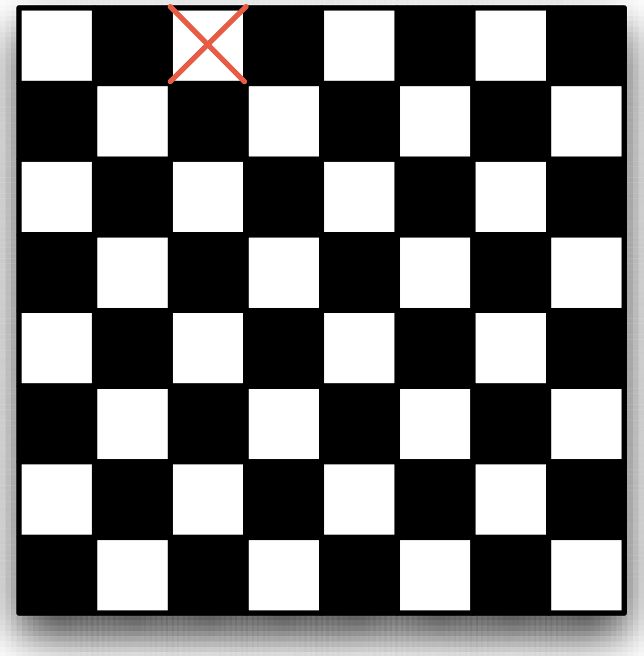


Goal: eliminate r90

$$\{q_i \mapsto j\} \in \mathrm{sol}(\mathrm{queens}) \Leftrightarrow \{q_j \mapsto n - i\} \in \mathrm{sol}(\mathrm{queens})$$

Example: Queens with SBDS

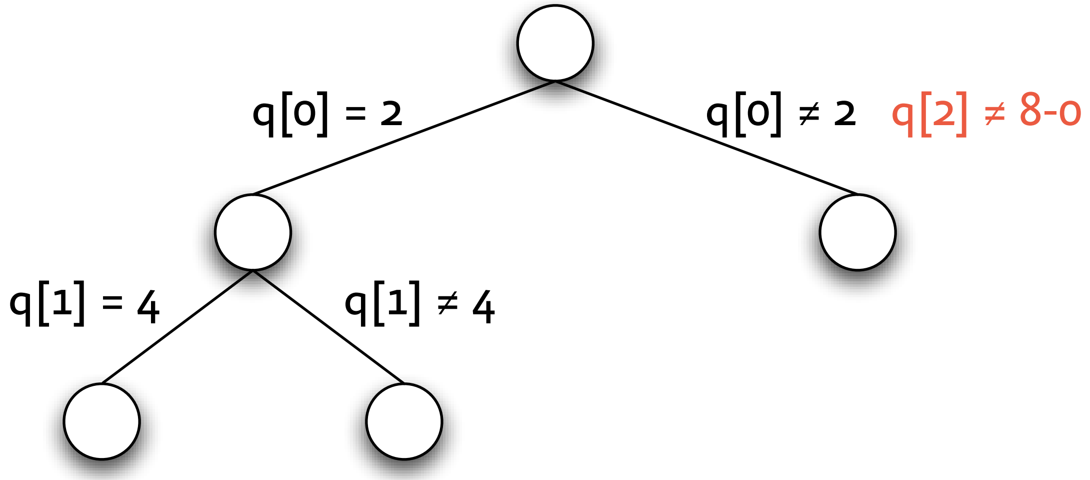


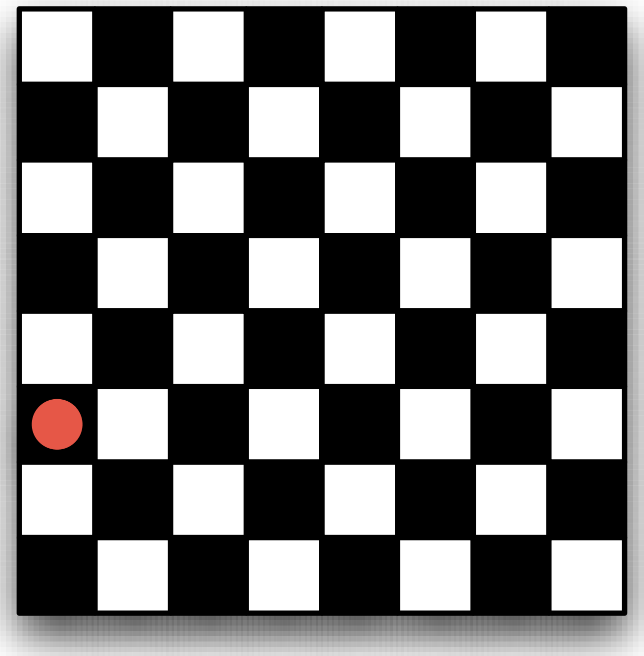


Goal: eliminate r90

$$\{q_i \mapsto j\} \in \mathrm{sol}(\mathrm{queens}) \Leftrightarrow \{q_j \mapsto n - i\} \in \mathrm{sol}(\mathrm{queens})$$

Example: Queens with SBDS

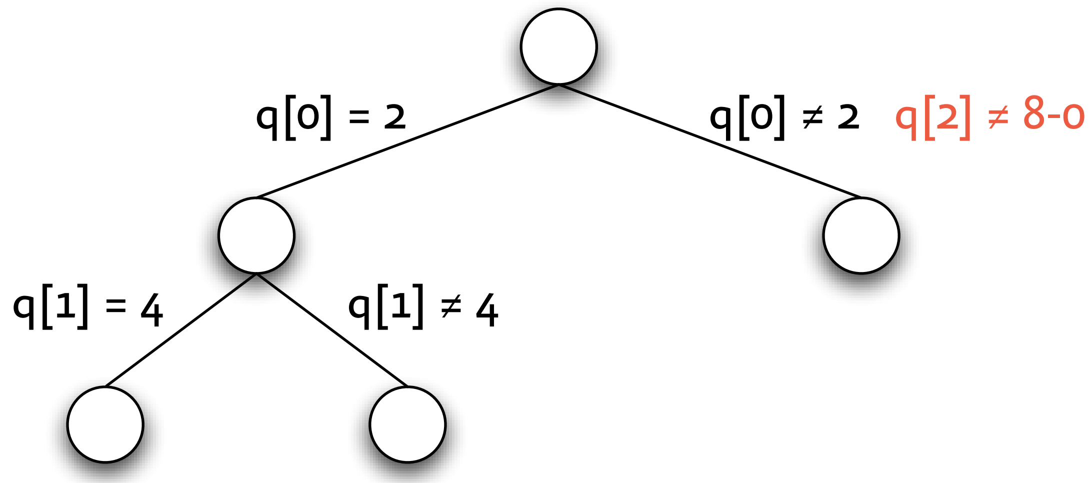


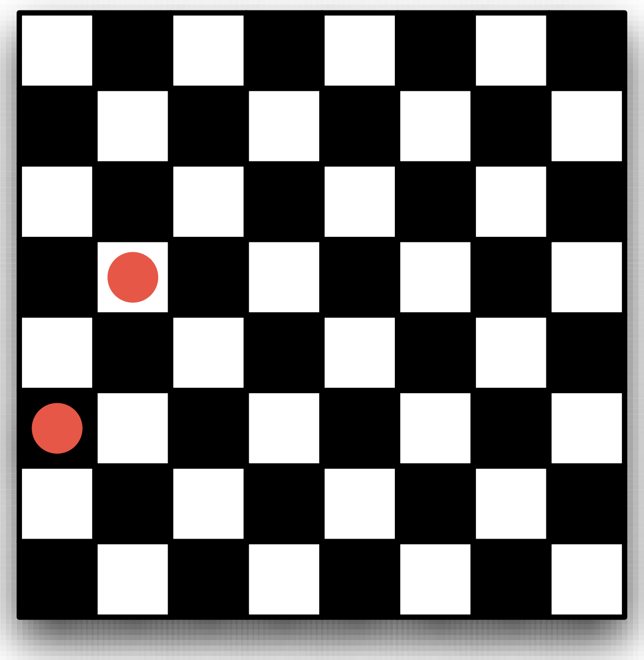


Goal: eliminate r90

$$\{q_i \mapsto j\} \in \mathrm{sol}(\mathrm{queens}) \Leftrightarrow \{q_j \mapsto n - i\} \in \mathrm{sol}(\mathrm{queens})$$

Example: Queens with SBDS

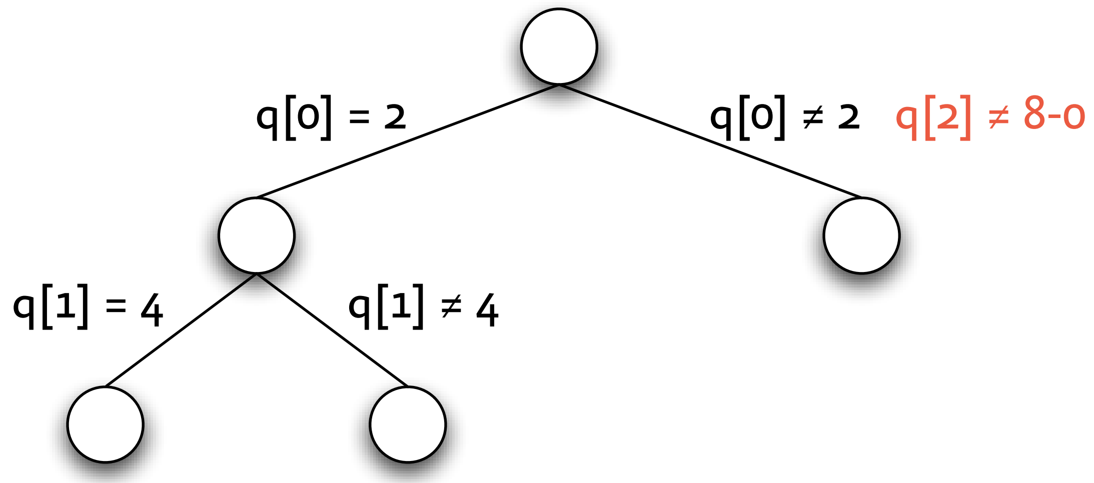


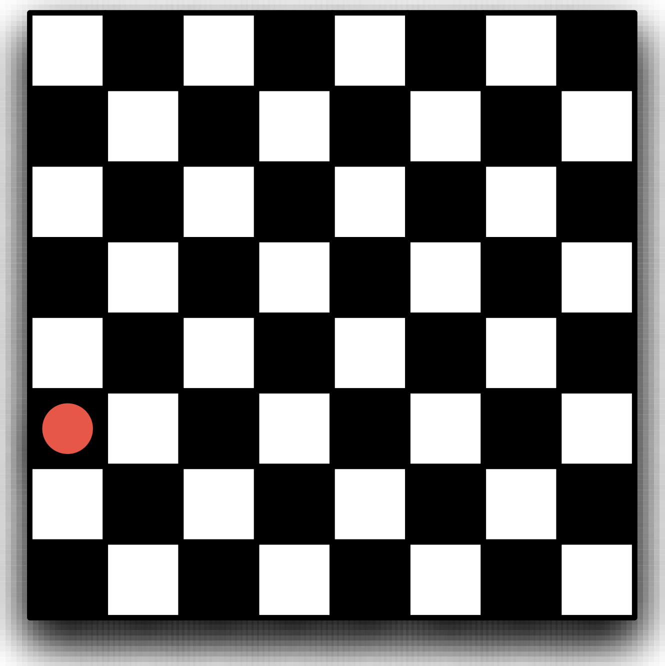


Goal: eliminate r90

$$\{q_i \mapsto j\} \in \text{sol}(\text{queens}) \Leftrightarrow \{q_j \mapsto n - i\} \in \text{sol}(\text{queens})$$

Example: Queens with SBDS

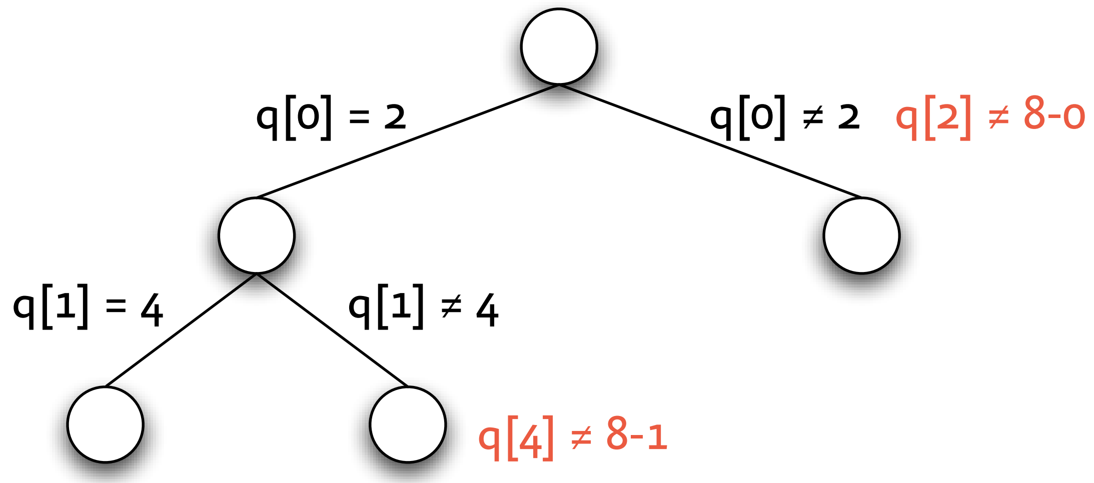


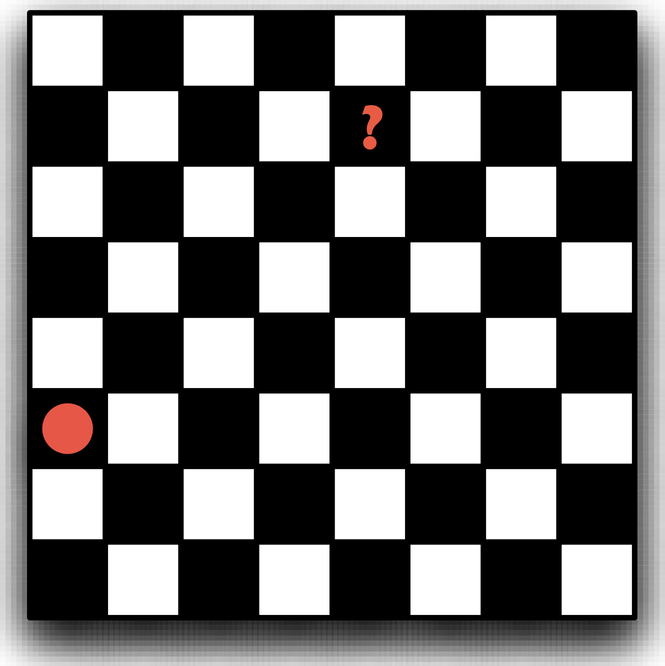


Goal: eliminate r90

$$\{q_i \mapsto j\} \in \mathrm{sol}(\mathrm{queens}) \Leftrightarrow \{q_j \mapsto n - i\} \in \mathrm{sol}(\mathrm{queens})$$

Example: Queens with SBDS

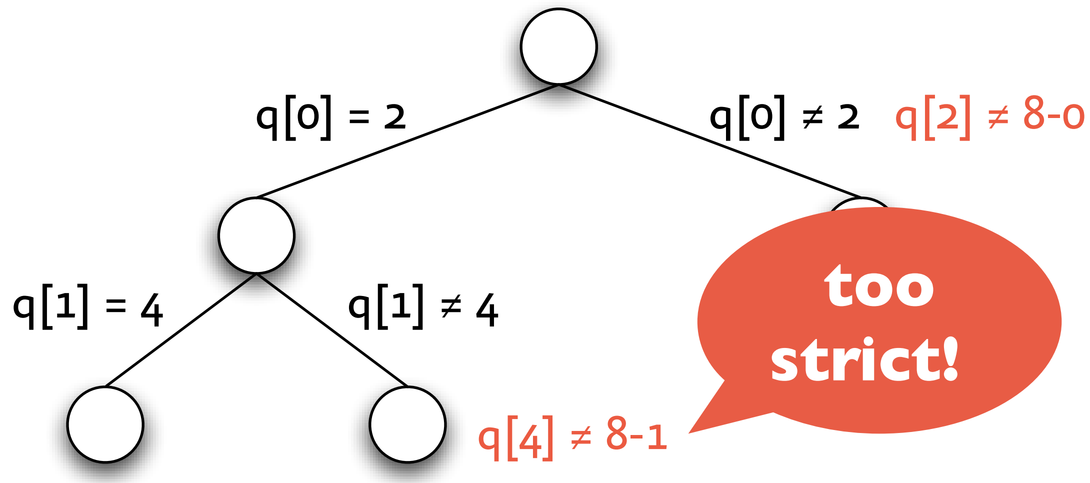


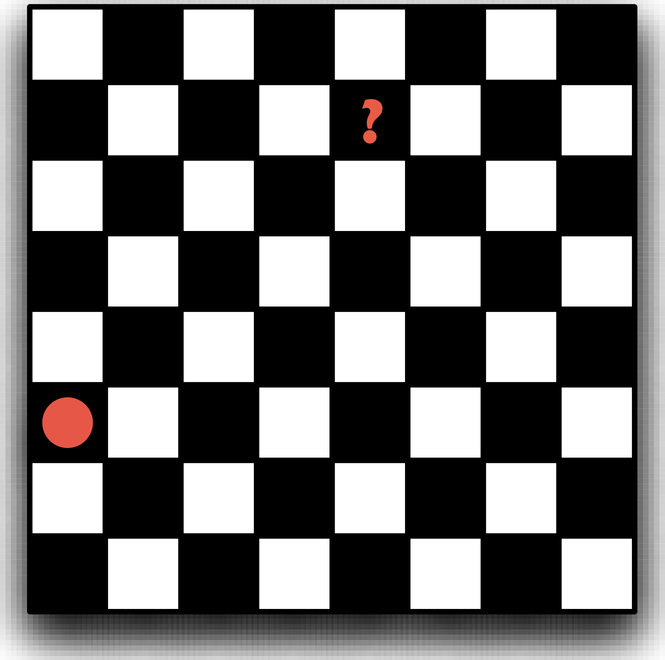


Goal: eliminate r90

$$\{q_i \mapsto j\} \in \mathrm{sol}(\mathrm{queens}) \Leftrightarrow \{q_j \mapsto n - i\} \in \mathrm{sol}(\mathrm{queens})$$

Example: Queens with SBDS

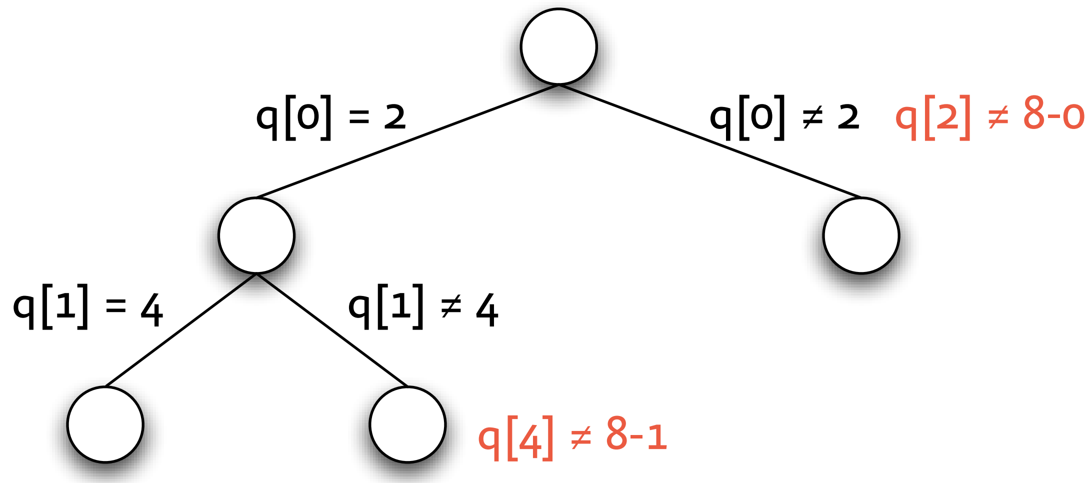


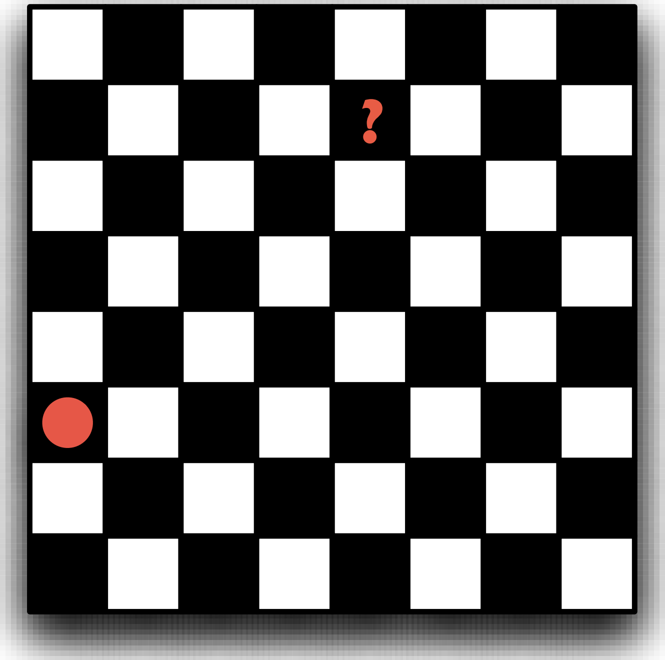


Goal: eliminate r90

$$\{q_i \mapsto j\} \in \text{sol}(\text{queens}) \Leftrightarrow \{q_j \mapsto n - i\} \in \text{sol}(\text{queens})$$

Example: Queens with SBDS

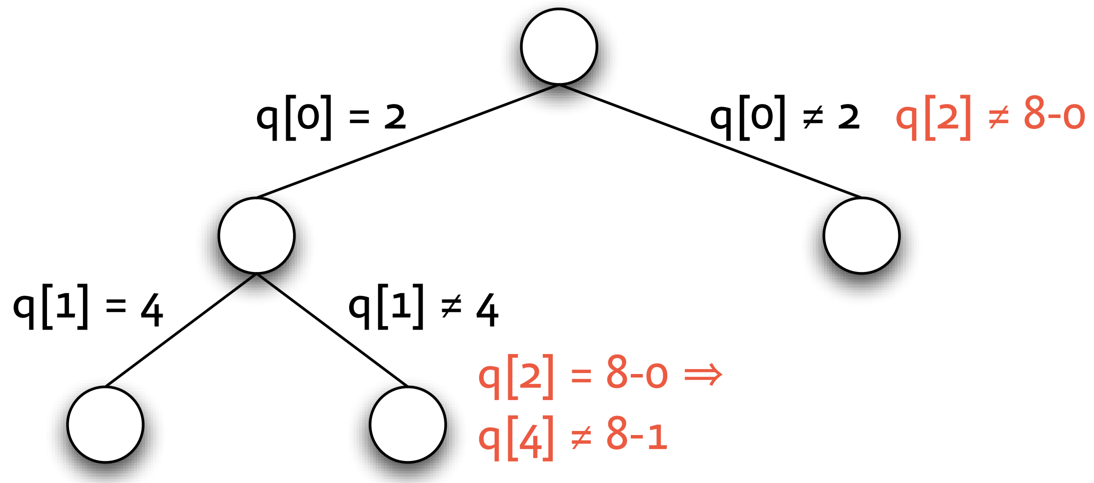


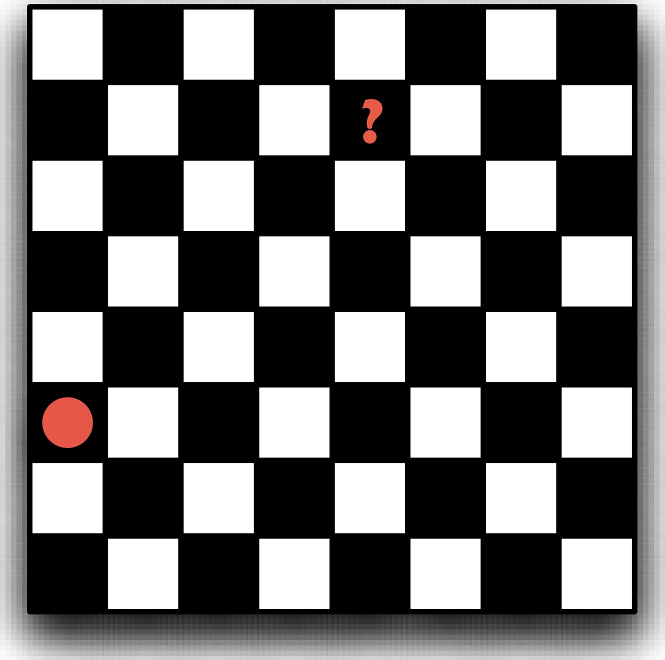


Goal: eliminate r90

$$\{q_i \mapsto j\} \in \mathrm{sol}(\mathrm{queens}) \Leftrightarrow \{q_j \mapsto n - i\} \in \mathrm{sol}(\mathrm{queens})$$

Example: Queens with SBDS





Goal: eliminate r90

$$\{q_i \mapsto j\} \in \mathrm{sol}(\mathrm{queens}) \Leftrightarrow \{q_j \mapsto n - i\} \in \mathrm{sol}(\mathrm{queens})$$

Implementation

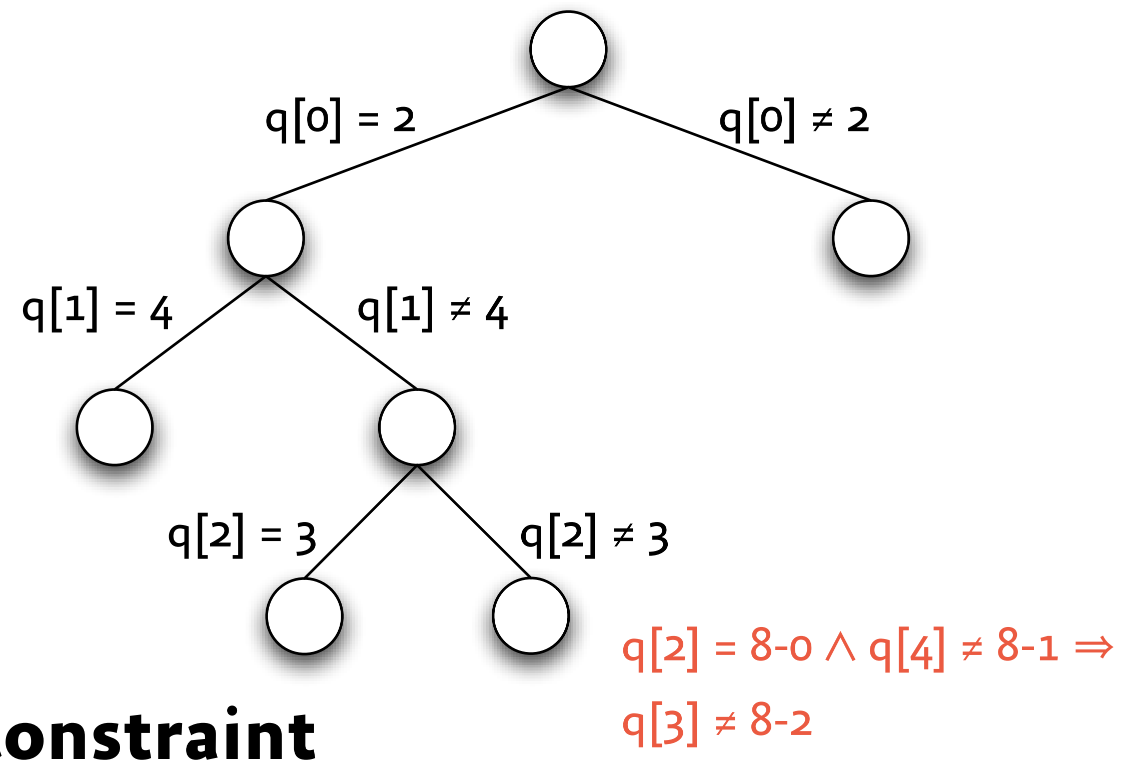
- **Collect prefix:**

  $(q[2] = 8) = b_1$

  $(q[4] \neq 7) = b_2$

  $b_1 \wedge b_2 \Rightarrow q[3] \neq 6$

- Use propagator for **reified constraint**



Disadvantages

- Can result in huge numbers of constraints being added (at each choice, one **for each symmetry**)
- All symmetries have to be specified explicitly

Literature

- Backofen, Will. *Excluding Symmetries in Constraint-Based Search*. Constraints 7(3), 2002.
- Gent, Smith. *Symmetry Breaking in Constraint Programming*. ECAI, 2000.

SBDD

(Symmetry Breaking by Dominance Detection)

Prune dominated subtrees

- **Idea:**

  if a search node is *dominated* by a node previously visited, don't descend

- Domination can be programmed
- No constraints added
- But previous states are memorized
- Similar concepts: nogoods, conflict clauses

SBDD ingredients

- **Dominance:**

  domain d_2 dominates d_1 iff $\forall x. d_1(x) \subseteq d_2(x)$

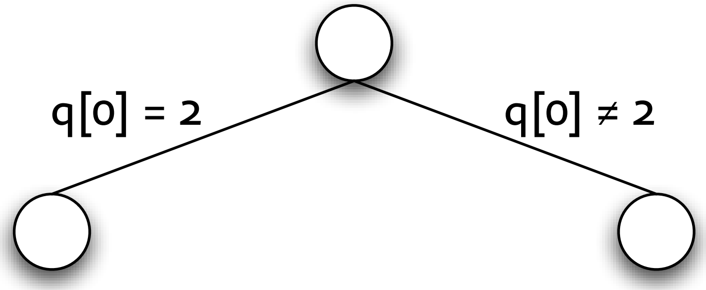
- **Detection:**

  function $\Phi : \mathrm{Dom} \times \mathrm{Dom} \rightarrow \mathbb{B}$

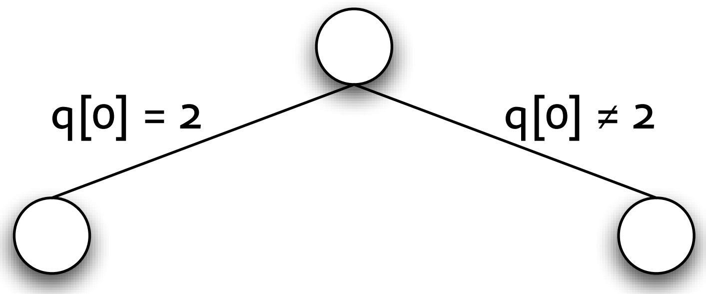
  such that $\Phi(d_1, d_2) = \mathrm{true}$ iff d_2 dominates d_1 under some symmetry σ

- **Database** T of already seen domains

Example: Queens with SBDD

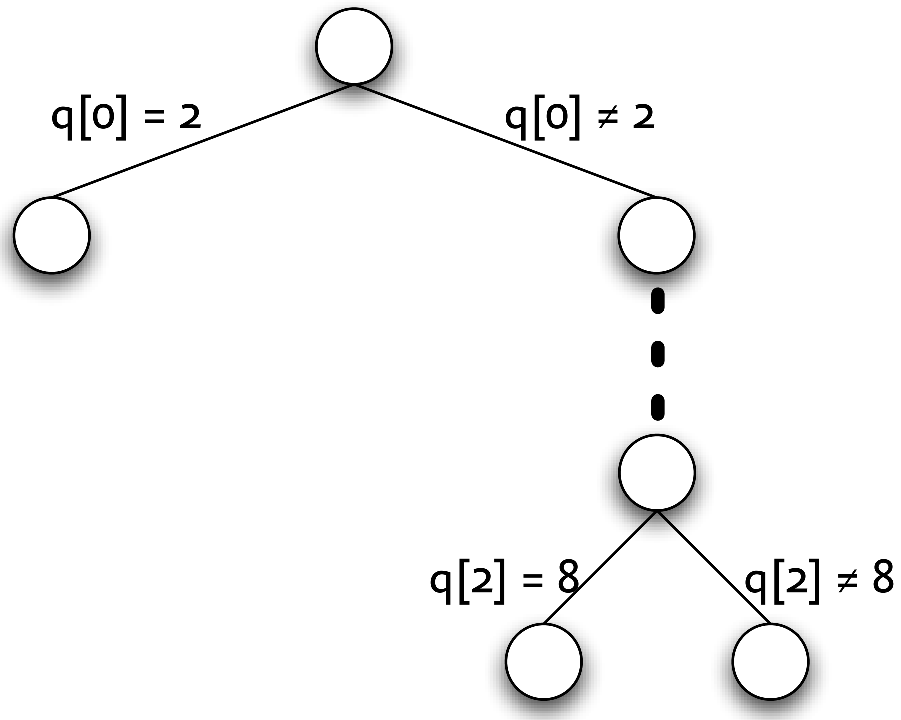


Example: Queens with SBDD



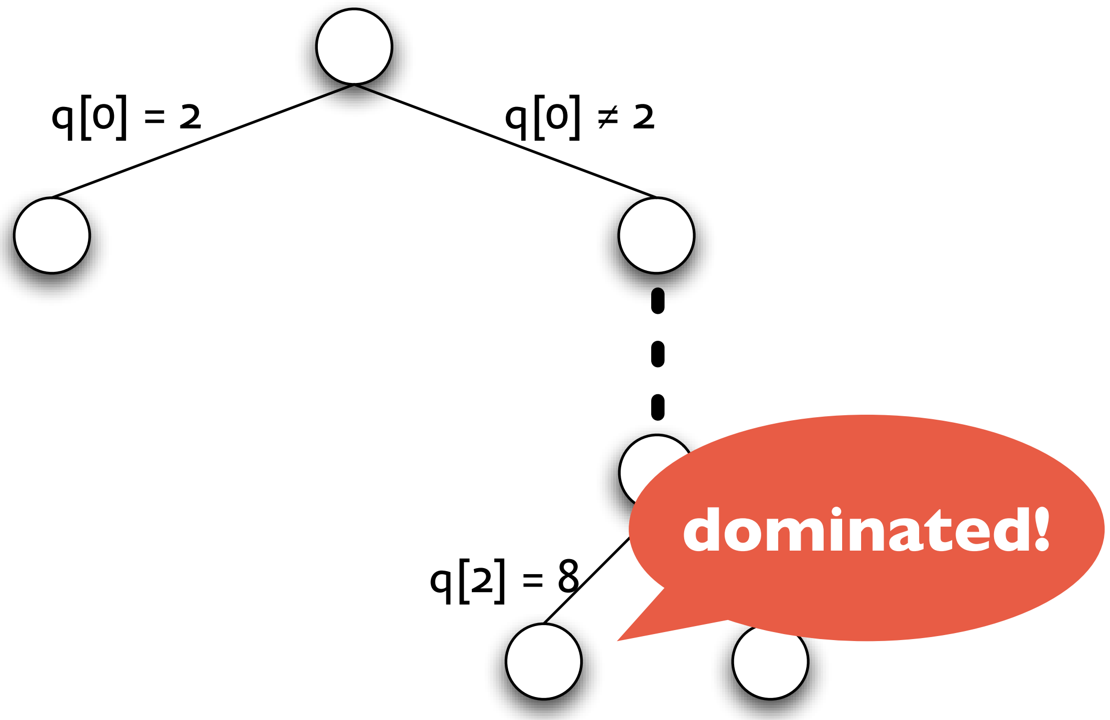
$T = \{ \{q[0]=2\} \}$

Example: Queens with SBDD



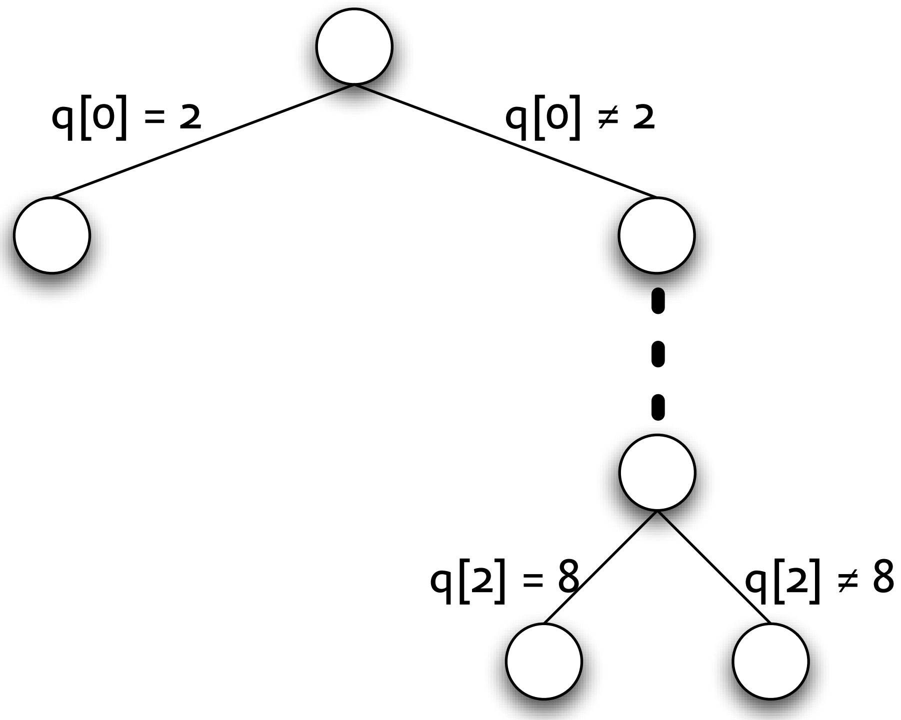
T = { {q[0]=2} }

Example: Queens with SBDD



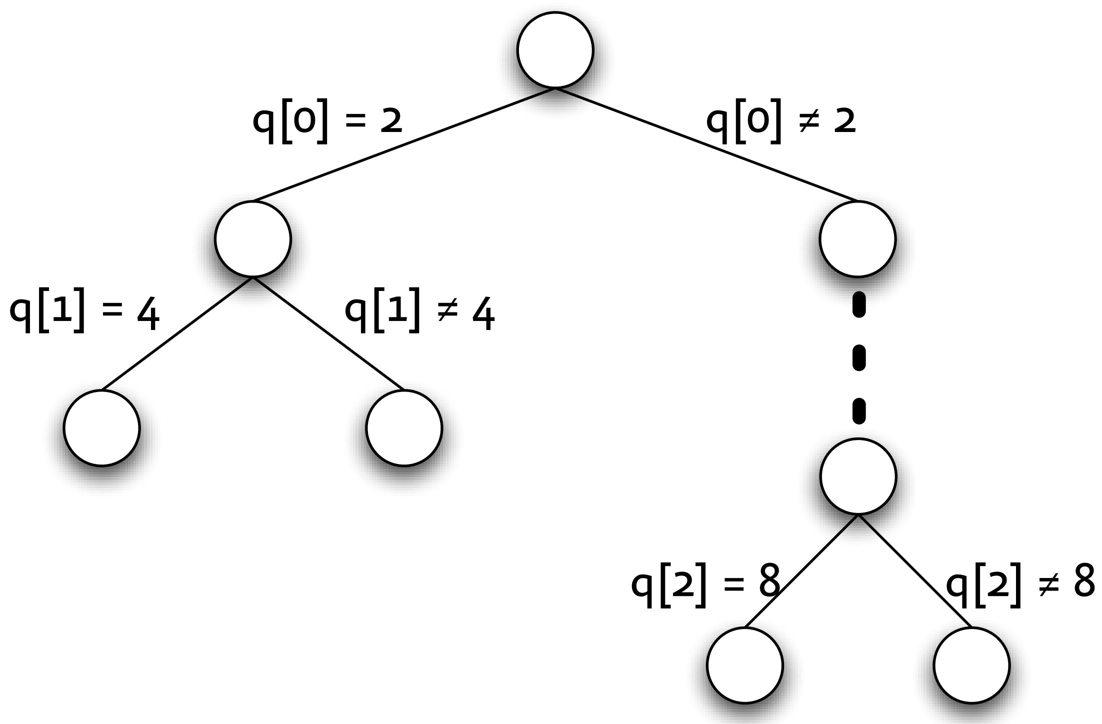
$T = \{ \{q[0]=2\} \}$

Example: Queens with SBDD



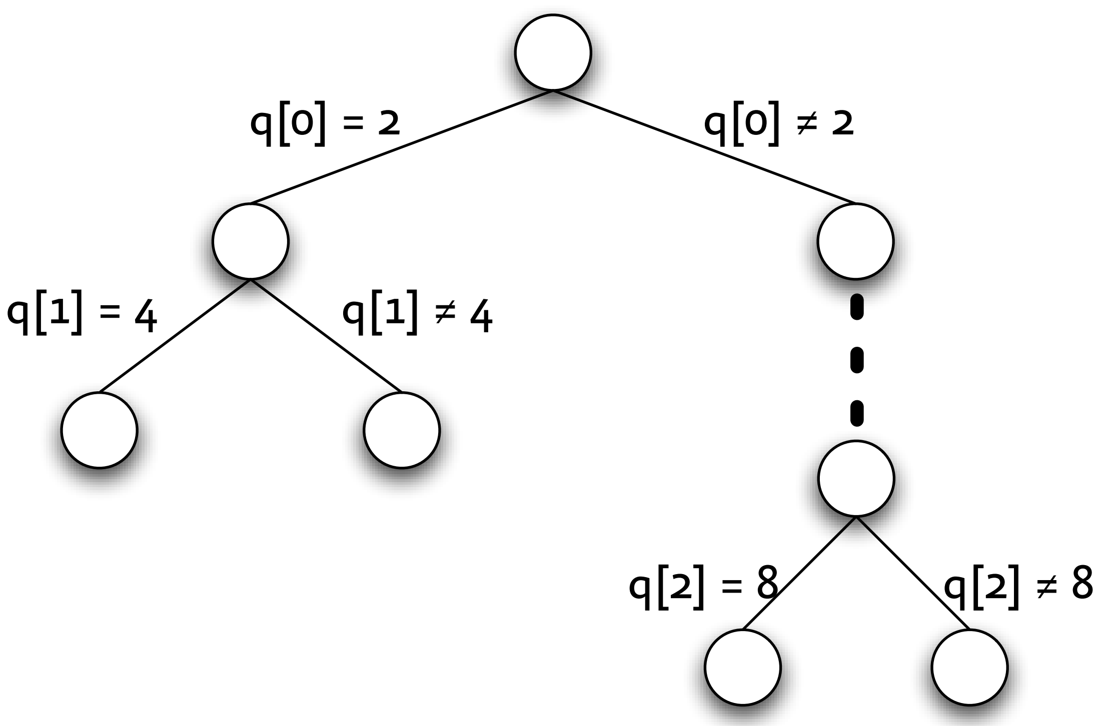
T = { {q[0]=2} }

Example: Queens with SBDD



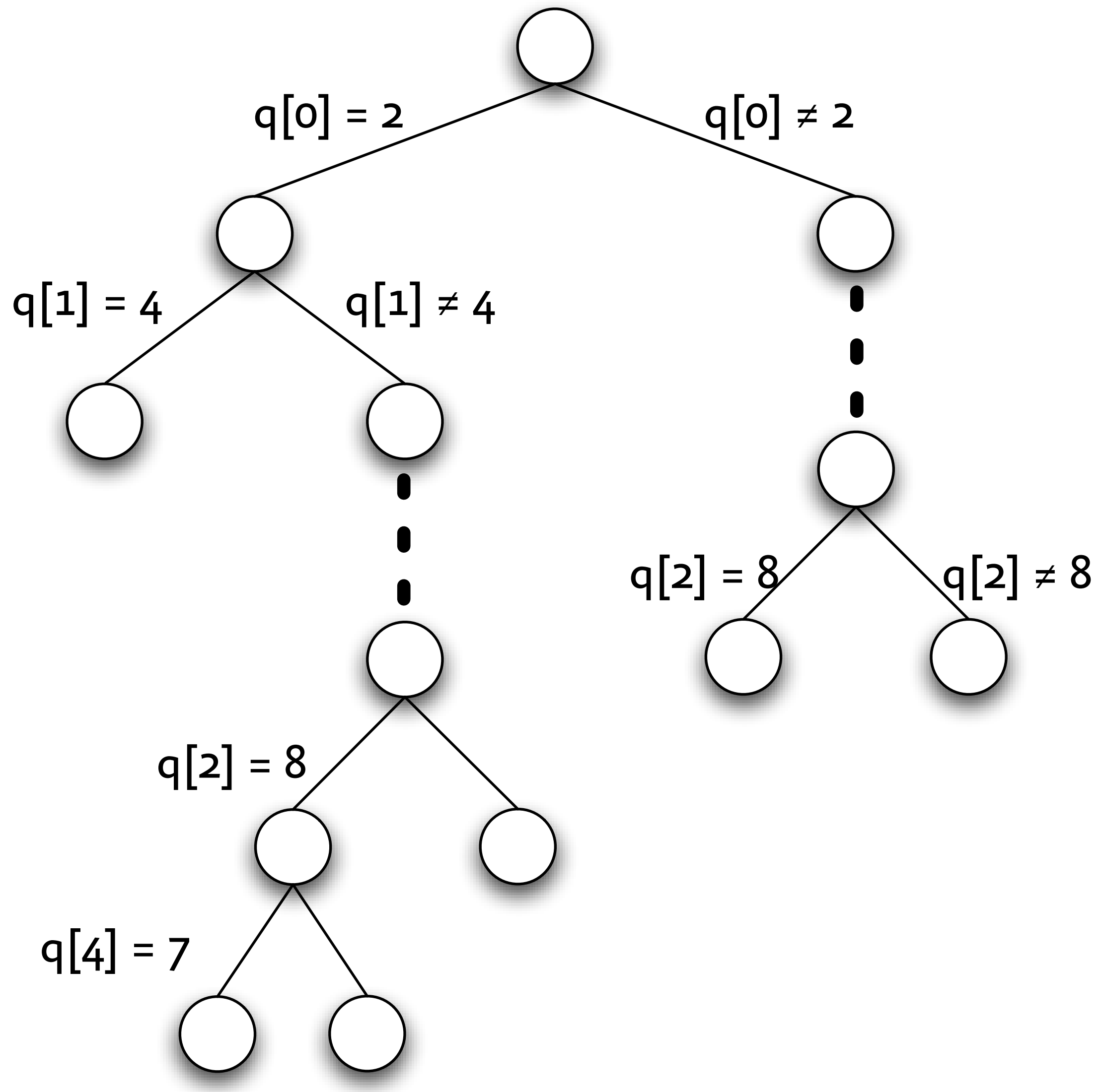
T = { {q[0]=2} }

Example: Queens with SBDD



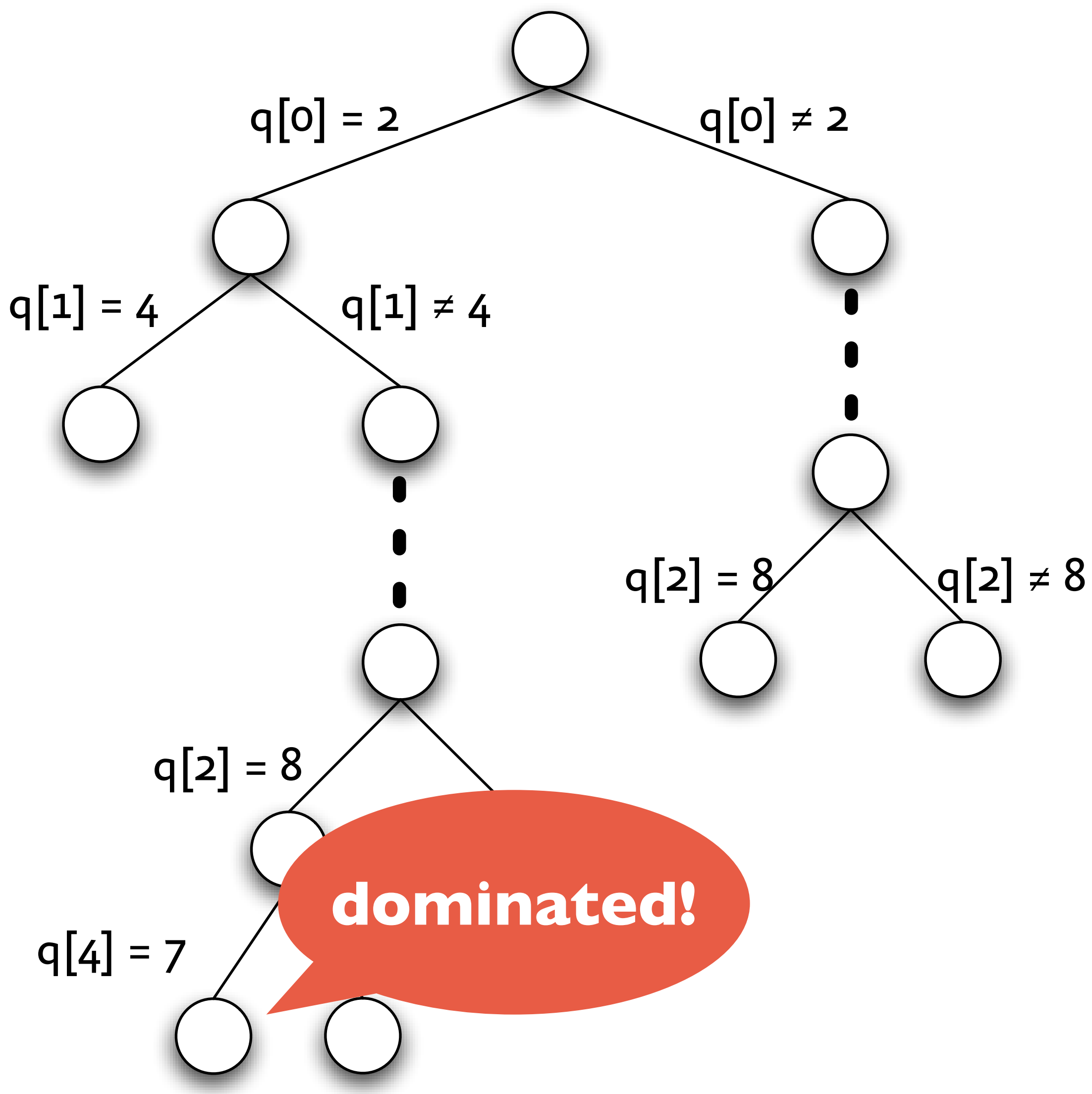
T = { {q[0]=2, q[1]=4} }

Example: Queens with SBDD



$T = \{ \{q[0]=2, q[1]=4\} \}$

Example: Queens with SBDD



T = { {q[0]=2, q[1]=4} }

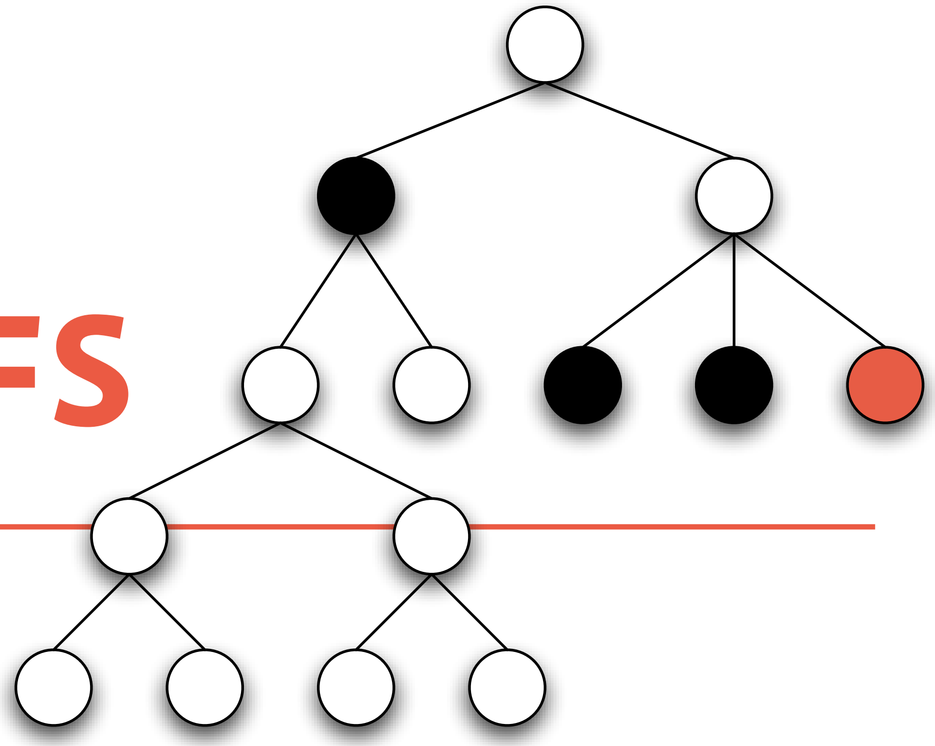
Optimization for DFS

- **Keeping all domains is infeasible**
- **Observation:**

  if d_2 is a successor of d_1, then d_1 dominates d_2
- **Optimization:**

  only keep domains left-adjacent to the path from the root to the current node

Optimization for DFS



- **Keeping all domains is infeasible**
- **Observation:**

  if d_2 is a successor of d_1, then d_1 dominates d_2
- **Optimization:**

  only keep domains left-adjacent to the path from the root to the current node

Using Φ for propagation

- **Derive a function**

  $$\text{prop}_\Phi(d_1, d_2, x) = \{v \in d_1(x) \mid \neg\Phi(d_1[x = v], d_2)\}$$

- If $\neg\Phi(d_1, d_2)$, use prop_Φ to prune domains of all x
- Prunes obviously domainted sub-trees

Pros and Cons

- **Good:** No constraints added
- **Good:** Handles all kinds of symmetry
- **Good:** Very configurable (by implementing Φ)
- **Bad:** Still all symmetries must be encoded
- **Bad:** Checking dominance at each node may be expensive

Literature

- Fahle, Schamberger, Sellmann. *Symmetry breaking*. CP, 2001.
- Sellmann, Van Hentenryck. *Structural Symmetry Breaking*. IJCAI, 2005.

Group theory

Reminder

- A group $(G, \times)$ is a set and an associated operation such that
  - G is closed under $\times$ — $i \times j \in G$
  - $\times$ is associative — $i \times (j \times k) = (i \times j) \times k$
  - G has an identity id — $i \times id = id \times i = i$
  - every element has an inverse — $i \times i^{-1} = i^{-1} \times i = id$

Permutation groups

- The set of **permutations** of a sequence forms a group
- **concatenation** is multiplication
  - closedness: $\sigma \bullet \sigma'$ is again a permutation
  - associativity
  - identity: $\sigma_{\mathrm{id}} = \{i \mapsto i\}$
  - inverse

Generators and orbits

- a set $S \subseteq G$ is called a **generator** of a group G iff
$$\forall g \in G\ \exists S' \subseteq S.\ g = \prod_{s \in S'} s$$
- the **orbit** of an element *i* w.r.t. a permutation group *G* is
$$O_G(i) = \{\sigma(i) \mid \sigma \in G\}$$

(can be extended to sets of points)

Using generators

- Generators describe groups **compactly**
- **Examples:**
  - symmetries of a square: < r90, d1 >
  - permutations of {1,...,n}: < (1,2,3,...,n),(1,2) >
- For variable or value symmetries: **easy**
- For variable/value symmetries: map pair (x_i,v) to $i|U|+v$
- Describe problem symmetries using generators

SBDS + group theory

- **Recall SBDS:**
  - for each symmetry g, post a constraint $g(A) \Rightarrow \neg g(c)$

    (for current partial assignment A and choice c)
  - only interested in **different** $g(A)$ and $g(c)$
  - compute the orbit of the current partial assignment A!

SBDD + group theory

- **basically:**

  a domain *d* in T dominates the current node *c* if *c* is in the orbit of *d*

- **more advanced:**

  use clever data structures and group theoretic algorithms

GAP

- **Groups, Algorithms, Programming**
- "A system for computational discrete algebra"
- http://www.gap-system.org

Literature

- Gent, Harvey, Kelsey. *Groups and Constraints: Symmetry Breaking during Search.* CP, 2002.
- Gent, Harvey, Kelsey, Linton. *Generic SBDD using GAP and ECLiPSe.* CP, 2003.

Summary

- **Symmetry is everywhere**
- Search enumerates **symmetric failure**
- **Possible cure:**
  - Model reformulation
  - Static symmetry breaking (lex-leader)
  - Dynamic symmetry breaking (SBDS, SBDD)
- Take advantage of **group theory**
  - compact specification of symmetries
  - algorithms