Pre-processing: Constraint propagation

- Node consistency: reduce the domain for unary constraints
- Arc consistency: every element in the domain satisfies binary constrains
- $x_i \in X_i : \exists x_j \in X_j$ such that (x_i, x_j) satisfies binary consistency.

Arc-consistency: AC-3 algorithm

Can arc-consistency guarantee global consistency?

- Why re-add (X_k, X_i)?
- Why not (X_i, X_k)?
- Time complexity?
- $\mathcal{O}(dn^2 \cdot d^2)$
- Remove-Inconsistent-Values
  - $\mathcal{O}(d^2)$

function AC-3(*csp*) **returns** the CSP, possibly with reduced domains
inputs: *csp*, a binary CSP with variables $\{X_1, X_2, \ldots, X_n\}$
local variables: *queue*, a queue of arcs, initially all the arcs in *csp*

while *queue* is not empty **do**
$(X_i, X_j) \leftarrow$ REMOVE-FIRST(*queue*)
if REMOVE-INCONSISTENT-VALUES(X_i, X_j) **then**
for each X_k **in** NEIGHBORS$[X_i]$ **do**
add (X_k, X_i) to *queue*
return *true*

---

function REMOVE-INCONSISTENT-VALUES(X_i, X_j) **returns** true iff succeeds
removed $\leftarrow$ *false*
for each *x* **in** DOMAIN$[X_i]$ **do**
if no value y in DOMAIN$[X_j]$ allows (*x*,*y*) to satisfy the constraint $X_i \leftrightarrow X_j$
then delete *x* from DOMAIN$[X_i]$; *removed* $\leftarrow$ *true*
return *removed*