AI – Inference in CSPs (http://www.it.uu.se/research/group/astra/CTcourse03/Slides/Consistency.pdf)

The key characteristic of CSPs is that they can *choose* the new variable assignment/successor to do a specific **constraint propagation**, voiding all non-legal variable assignments as we go on

***Node consistency***

**A variable is node-consistent if all the values in the variable’s domain satisfy the variable’s unary constraint**

e.g. The map coloring problem, having SA≠green and the domain {red, green, blue}, we remove green from SA’s domain

***Arc consistency***

**A variable is arc consistent if every value in its domain satisfies the variable’s binary constraints**

X is arc-consistent wrt Y if ∀ value in the domain DX ∃ some value in the domain DY that satisfies the binary constraint on the arc (X, Y)

This also involves domain pruning

**A CSP (or graph) is arc-consistent iff all variables are arc-consistent with one another**

**REM:**

* CSPs with any arity are reducible to binary CSPs

***The AC-3 algorithm*** (implementation in book)

AC-3 is used to enforce arc consistency

Initially, the queue contains all the arcs in the CSP. (Each binary constraint becomes two arcs, one in each direction.)

AC-3 then pops off an arbitrary arc (Xi, Xj) from the queue and makes Xi arc-consistent with respect to Xj

**If this leaves Di unchanged, the algorithm just moves on to the next arc**; but if this revises Di (makes the domain smaller), then we add to the queue all arcs (Xk, Xi) where Xk is a neighbor of Xi

We need to do that because the change in Di might enable further reductions in Dk, even if we have previously considered Xk

**If Di is revised down to nothing, then we know the whole CSP has no consistent solution, and AC-3 can immediately return failure**

Otherwise, we keep checking, trying to remove values from the domains of variables until no more arcs are in the queue; this is O(cd³), c: number of constraints, d: domain size

***Path consistency***

**Path consistency tightens the binary constraints by using implicit constraints inferred by looking at triples of variables**

A two-variable set {Xi, Xj} is path-consistent with respect to a third variable Xm if, ∀assignment {Xi = a, Xj = b} consistent with the constraints (if any) on {Xi ,Xj}, ∃ an assignment to Xm that satisfies the constraints on {Xi, Xm} and {Xm, Xj}

Trivially, the variable Xm is in the middle of the path from Xi to Xj

***K-consistency*** (EXPTIME/EXPSPACE problem)

**A CSP is k-consistent if, for any set of k-1 variables and for any consistent assignment to those variables, a consistent value can always be assigned to any kth variable**

1-consistency is node consistency (given the empty set, we can make any set of one variable consistent)

2-consistency is arc consistency

3-consistency is path consistency

**A CSP is strongly k-consistent if it’s k-consistent up to k=1**

k-consistency does not imply k-1-consistency