Consistency - Chapter 5

- Introduce several notions of Local Consistency:
  - arc consistency,
  - hyper-arc consistency,
  - k-consistency and strong k-consistency.
- Local consistency is about the existence of partial solutions and their extensions.
- Local consistency helps search by making partial solutions easier to find.

Example

Consider the three integer variables x, y and z each with the domain $\{0, 1, \ldots, 10\}$ and the single constraint:

$$x + y = z$$

For any variable and any value there is a solution containing that value and variable.

Example Continued

$x, y, z \in \{0, 1, \ldots, 10\}$ and $x + y = z$.

- For any value, x, of X set Y to be 0 and Z to be x;
- For any value, y, of Y set X to be 0 and Z to be y;
- For any value, z, of Z set X to be 0 and Y to be z.

But if we set X to be 8 then this imposes the restrictions:

- $0 \leq Y \leq 2$
- $8 \leq Z \leq 10$.

Local Consistency

- The various notions of local consistency help understand what is going on in the previous example.
- Constraint propagation attempts to reduce domains so that in the previous example assigning 8 to X reduces the domain of Y and Z so that during search fewer possibilities have to be explored.
- Propagation is the heart of modern constraint programming. Without it constraint programming just reduces to generate and test.

Arc Consistency of a Binary Constraint

- A binary constraint C on the variables x and y with domains X and Y is a subset of $X \times Y$. Such a $C \subseteq X \times Y$ is *arc consistent* if:
  - $\forall a \in X$ there $\exists b \in Y$ such that $(a, b) \in C$;
  - $\forall b \in Y$ there $\exists a \in X$ such that $(a, b) \in C$.
- Note both directions are important.
- A CSP is said to be *consistent* is all its binary constraints are consistent (this definition is unproblematic if all the constraints in the CSP are binary).

Examples of Arc-consistency

- For $x \in [2 \ldots 6]$, $y \in [3 \ldots 7]$ the constraint

$$x < y$$

  is arc-consistent.

- For example
  - if $x = 2$ then there is a solution
  - if $x = 6$ then y must be 7
  - if $y = 3$ then x must be 2.

A Non Arc-Consistent Constraint

- For $x \in [2..7]$ and $y \in [3..7]$ the constraint:

$$x < y$$

  is not arc consistent.

- If $x = 7$ (allowed by the domains) then there is no value for y satisfying the constraint.

Status of Arc Consistency

- Arc consistency does not imply consistency. Given $x \in \{a, b\}$ and $y \in \{a, b\}$ the two constraints

  $$x = y \text{ and } x \neq y$$

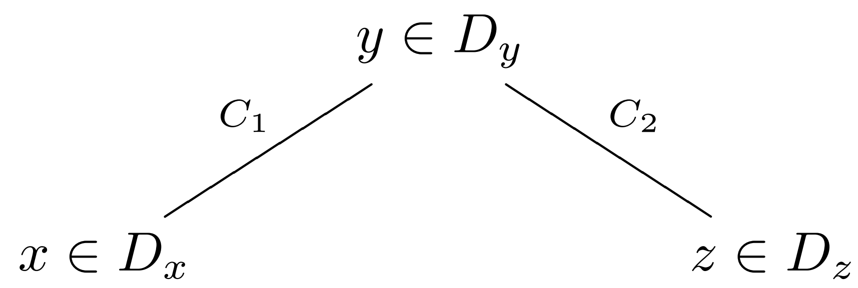
  are both arc-consistent.

- but there is obviously no solution to *both* constraints.

Status of Arc Consistency

For particular CSPs arc consistency implies consistency.

- Given a CSP



  where each constraint is arc-consistent, the whole CSP is consistent.

- To see this pick a value for y then arc-consistency gives a value for x and z.

In general if the constraint graph is a tree then arc consistency implies consistency.

Achieving and Using Arc-Consistency

- The Basic Idea
  - remove values from the domain which do not take part in solutions.
- Given a constraint $C \subseteq D_x \times D_y$ on the variables $x \in D_x$ and $y \in D_y$ reduce the domains by the following two rules:
  - $D'_x = \{a \in D_x \mid \exists b \in D_y \text{ s.t. } (a,b) \in C\}$
  - $D'_y = \{b \in D_y \mid \exists a \in D_x \text{ s.t. } (a,b) \in C\}$
- To achieve arc consistency you must apply both rules.

Directional Arc Consistency

- Sometimes it can be expensive (computationally) to achieve arc consistency: perhaps because the domains are large.
- While searching for values of variables with backtrack search you might not need full arc consistency.
- Suppose you always assign a value to x before you assign a value to y then given a constraint C on x and y all you need is that for all values, a, of x there is a tuple (a, b) in C giving a value for y.

This leads to the notion of directional consistency.

Directional Arc Consistency

Ingredients:

- A linear order $\preceq$ on the variables.
- A linear order is a binary relation such that:
  - For all x, $x \preceq x$ (reflexivity)
  - For all x and y, $x \preceq y$ and $y \preceq x$ implies $x = y$ (antisymmetry)
  - For all x,y and z, $x \preceq y$ and $y \preceq z$ implies $x \preceq z$ (transitivity)
  - For all x and y either $x \preceq y$ or $y \preceq x$.
- A linear order is just some fixed order on the variables.

Directional Arc Consistency

Assume a linear ordering $\preceq$ on the variables:

- A constraint C on $x \in D_x$, $y \in D_y$ is directionally arc consistent w.r.t. $\preceq$ if:
  - $\forall a \in D_x$ there $\exists b \in D_y$ such that $(a, b) \in C$ provided that $x \preceq y$.
  - $\forall b \in D_y$ there $\exists a \in D_x$ such that $(a, b) \in C$ provided that $y \preceq x$.
  - A CSP is directionally arc consistent w.r.t. $\preceq$ if all its binary constraints are.

Directionally Arc Constraints - Example

- Given $x \in [2 \ldots 7]$, $y \in [3 \ldots 7]$ the constraint $x < y$ is not arc consistent (no solution for $x = 7$).
- It is directionally arc consistent w.r.t. $y \preceq x$. That is for any value you assign to y there is an assignment to x satisfying the constraint.
- Is is not directionally consistent when $x \preceq y$: for example assigning 7 to x means that we can not assign any value to x.

Non Binary Constraints and Consistency

- Most constraints that you meet are not binary, for example you have already met the `alldifferent` constraint.
- Although you can always model a problem with binary constraints it is not always as efficient as using global (non-binary) constraints.
- There are lots of possible definitions of arc-consistency possible for non-binary constraints.
- We will look at a pragmatically useful one which is often used in implementations.

Hyper-Arc Consistency

- A constraint on the variables $x_1, \ldots, x_n$ with the domains $D_1, \ldots, D_n$ is *hyper-arc consistent* if

$$\forall i \in [1..n] \forall a \in D_i \text{ there } \ \exists d \in C \text{ s.t. } a = d[x_i]$$

- The notation $d[x_i]$ takes a tuple of values in a relation and projects it down to the entry corresponding to the variable x_i.
- The tuple d is often called a supporting tuple of the assignment $x = a$.

Hyper-arc Consistency - Examples

Suppose C is a constraint on the variables $x \in \{1,2,3\}$, $y \in \{1,2,3\}$ and $z \in \{1,2,3\}$.

Suppose $C(x,y,z)$ is given by a list of tuples: $\langle 1,2,3\rangle$,$\langle 1,2,2\rangle$ and $\langle 2,3,3\rangle$ then this is not hyper-arc consistent w.r.t to the domains since for $z = 1$ there is no tuple supporting it.

The first constraint in this lecture is hyper-arc consistent ($x, y, z \in \{0, 1, \ldots, 10\}$ and $x + y = z$).

Global Constraints

- Given some global constraints how to prune values from the domain to keep hyper-arc consistency.
- Problem to do this efficiently.
- Later on in the course you will meet many global constraints and pruning algorithms.

Instantiations

Fix a CSP $\mathcal{P}$.

- An instantiation is a function on a subset of the variables of $\mathcal{P}$ which assigns a value in the domain.
- Notation from the book:

$$\{(x_1, d_1), \ldots, (x_n, d_n)\}$$

  means x_1 is assigned to d_1, …, x_n is assigned to d_n.
- Another common notation:

$$x_1 \mapsto d_1 \wedge \cdots \wedge x_n \mapsto d_n$$

Consistent Instantiations

We want a notion of when a partial solution satisfies a CSP $\mathcal{P}$.

- Given an instantiation I on the variables X, we denote the restriction of I to a set $Y \subset X$:

$$I|Y$$

- An instantiation I with domain X is *consistent* if for every constraint C of $\mathcal{P}$ on the variables Y with $Y \subset X$, $I|Y$ satisfies C. (I will often refer to consistent instantiations as partial solutions).
- A Consistent instantiation is a k-*consistent* instantiation if its domain consists of k variables.

Example

Let $\mathcal{P}$ be the CSP

$$x < y \ , \ y < z \ , \ x < z \ ; \ x \in [0 \ldots 4] \ , \ y \in [1 \ldots 5] \ , \ z \in [5 \ldots 10]$$

Let I be $x \mapsto 0 \wedge y \mapsto 5 \wedge z \mapsto 6$

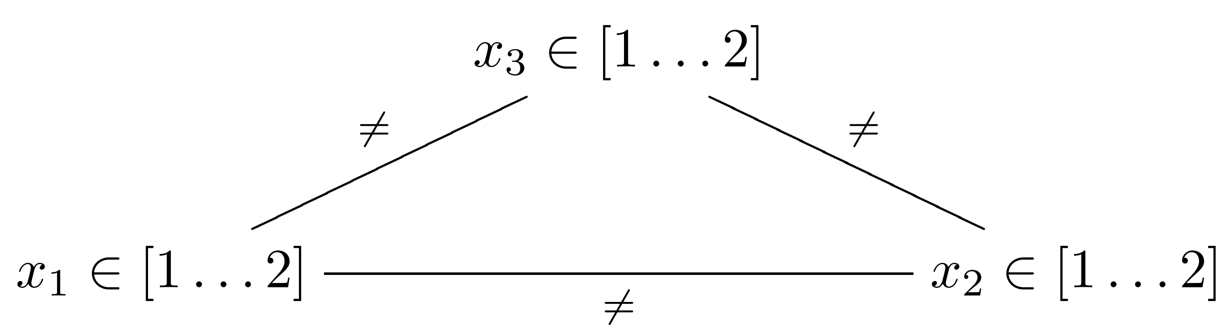
- $I|\{x,y\} = x \mapsto 0 \wedge y \mapsto 5$ and satisfies $x < y$;
- $I|\{x,z\} = x \mapsto 0 \wedge z \mapsto 6$ and satisfies $x < z$;
- $I|\{y,z\} = y \mapsto 5 \wedge z \mapsto 6$ and satisfies $y < z$.
- So I is a 3-consistent instantiation. It is a solution to the CSP.

k-Consistency

- A CSP is 1-*consistent* if for every variable x every unary constraint equals the domain of x (Node consistency).
- A CSP is k-*consistent* for $k > 1$ if every $k-1$-consistent instantiation can be extended to a k-consistent instantiation *no matter* which new variable is chosen.

Example

Consider the CSP.



This is 2-consistent.

Pick any partial solution:

$$x_1 \mapsto 1$$

pick any other variable say x_3 and we can find a consistent instantiation satisfying the constraints say $x_3 = 2$.

Example Continued

But

$$\begin{tikzcd} & x_3 \in [1 \ldots 2] \arrow[dl, dash, "\neq"'] \arrow[dr, dash, "\neq"] & \\ x_1 \in [1 \ldots 2] \arrow[rr, dash, "\neq"'] & & x_2 \in [1 \ldots 2] \end{tikzcd}$$

is not 3-consistent.

To prove this we pick some 2-consistent partial solution:

$$x_1 \mapsto 1 \wedge x_2 \mapsto 2$$

this cannot be extended to a solution on 3 variables.

Recap

- To show k consistency you have to look at every consistent $k-1$ solution and every other variable and show that the extension exists.
- To disprove consistency you only have to find one counter example.

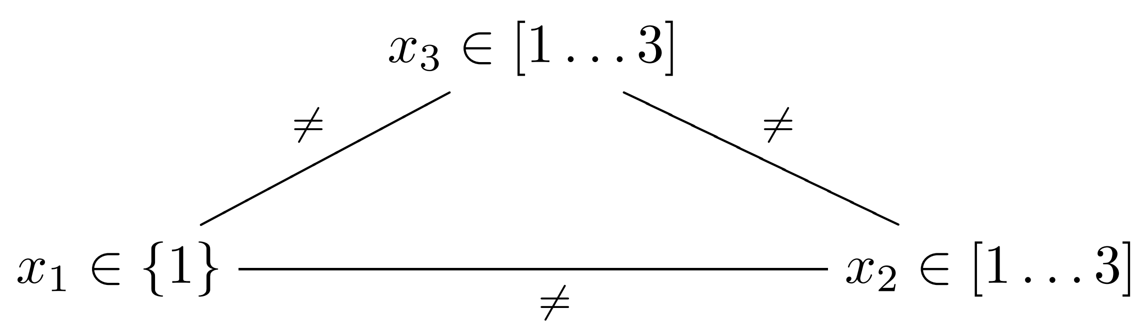
Why is Consistency a Good Thing?

- If your CSP is k consistent you know that if you have assigned $k-1$ variables you can assign the next variable in the search tree.

But k-consistency is not the whole story.

- k consistency does not imply $k-1$ consistency (example next slide);
- If your CSP is k consistent you still have to find a $k-1$ consistent partial solution.

Example



This CSP is consistent but not 3 consistent. There is a solution:

$$x_1 \mapsto 1 \wedge x_2 \mapsto 2 \wedge x_3 \mapsto 3$$

but the partial solution:

$$x_3 \mapsto 1 \wedge x_2 \mapsto 2$$

has no extension.

Strong k-Consistency

- A CSP is *strongly k-consistent* where $k \geq 1$ if it is i-consistent for every $i \in [1 \ldots k]$.
- A CSP with non-empty domains on k variables which is k-consistent has a solution.
- Sometimes we can do better (later in the course) and show when we only need a certain amount of local consistency to achieve global consistency.
- In the next lecture we will see that it is possible to make a CSP k-consistent (or show it is not consistent) for any k. So one way of solving the CSP is to progressively increase the level of consistency until the CSP is solved.

Path Consistency

Path Consistency is a special case of k-consistency when $k = 2$ (plus some other conditions to be spelled out below).

Consider the CSP

$$x < y \ , \ y < z \ , \ z < x$$

with $x \in \{1 \dots 1000\}$, $y \in \{1 \dots 1000\}$ and $z \in \{1 \dots 1000\}$.

This can be shown to be inconsistent by using the arc-consistency domain reduction rules. Applying the rule to $x < y$ gives that $x \in \{1 \dots 999\}$ then applying the rule $z < x$ gives $z \in \{1 \dots 998\}$ and so on. Eventually you can show that the CSP is inconsistent.

Path Consistency

But the CSP

$$x < y \ , \ y < z \ , \ z < x$$

can be shown to be inconsistent: by combining the two constraints $x < y$ and $y < z$ allows you to deduce that $x < z$ contradicting the constraint $z < x$.

Operations on Binary Relations

Path consistency generalises the previous example to arbitrary binary constraints.

Given two binary relations R and S define the following operations:

- the *transpose* of R by

  $$R^T = \{(b, a) \mid (a, b) \in R\}$$

  some people write R^{op} instead of R^T;

- the composition of R and S by

  $$R \cdot S = \{(a, c) \mid \exists b \text{ s.t. } (a, b) \in R \wedge (b, c) \in S\}$$

Examples of Composition

- Let

$$R = \{(1,2),(1,3),(4,3),(2,3)\}$$

- and

$$S = \{(1,3),(3,1),(4,2)\}$$

- then

$$R \cdot S = \{(1,1),(4,1),(2,1)\}$$

- and

$$S \cdot R = \{(3,2),(3,3),(4,3)\}$$

Normalised CSPs

A CSP is *normalised* if for every subsequence x,y of variables at most one constraint on x, y exists (defn. 5.15)

The CSP

$$x + y < 5,\ x + y \neq 2,\ x \in \{0 \ldots 4\},\ y \in \{0 \ldots 4\}$$

is not normalised.

A CSP can be made normalised by taking the conjunctions of the multiple constraints.

Path Consistency

Notation a C constraint on the variables x, y will be denoted $C_{x,y}$.

Given a constraint on the variables x, y there is also an imaginary constraint $C^T_{y,x}$ on the variables y, x which is the transpose of the constraint C.

A normalised CSP is *path consistent* (Defn. 5.18) if for every subset of variables $\{x, y, z\}$ of its variables have

$$C_{x,z} \subseteq C_{x,y} \cdot C_{y,z}$$

Example

$$\begin{tikzcd}
& y \in [1 \ldots 5] \arrow[dl, dash, "x<y"'] \arrow[dr, dash, "y<z"] & \\
x \in \{0 \ldots 4\} \arrow[rr, dash, "x<z"'] & & z \in [6 \ldots 10]
\end{tikzcd}$$

Path Consistency

- To make a CSP arc-consistent we reduced the domains.
- To make it path-consistent we reduce the constraints.
- Path consistency is about triangles of relations, so there are three rules.

Path Consistency Rules

- Given $C_{x,y}$,$C_{x,z}$,$C_{y,z}$ replace $C_{x,y}$ by

$$C'_{x,y} = C_{x,y} \cap (C_{x,z} \cdot C^T_{y,z})$$

- Given $C_{x,y}$,$C_{x,z}$,$C_{y,z}$ replace $C_{x,z}$ by

$$C'_{x,z} = C_{x,z} \cap (C_{x,y} \cdot C_{y,z})$$

- Given $C_{x,y}$,$C_{x,z}$,$C_{y,z}$ replace $C_{y,z}$ by

$$C'_{y,z} = C_{y,z} \cap (C^T_{x,y} \cdot C_{x,z})$$