ECE570 Lecture 11: Arc Consistency

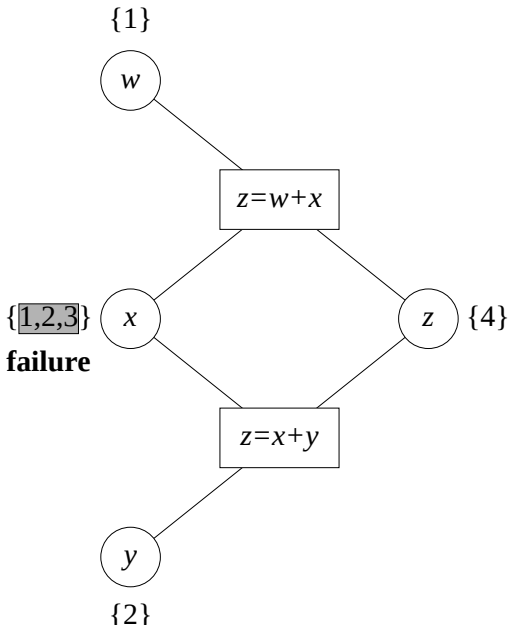
Jeffrey Mark Siskind

School of Electrical and Computer Engineering

Fall 2013

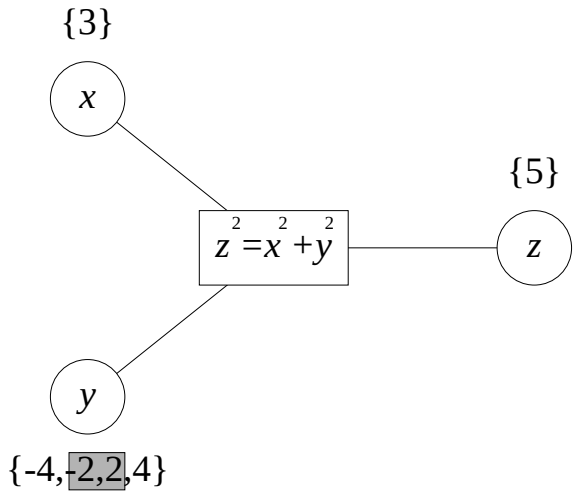
PURDUE UNIVERSITY

Cascaded Inference Leading to Failure



Failure is not detected by Forward Checking.
Cascaded inference can lead to failure.
Inference order can affect efficiency.

Generalized Forward Checking

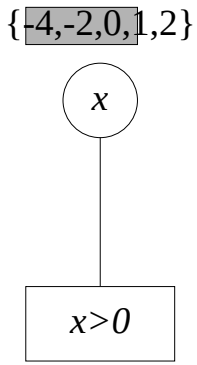


If there exists some constraint such that

1. all but one of its arguments are bound

then remove those elements from the domain of the unbound argument that are inconsistent with the remaining bound arguments.

Node Consistency—I



A vertex x_i in a CSP hypergraph is consistent if for each unary constraint P applied to x_i:

$$(\forall x \in D_i)P(x)$$

A CSP is *node consistent* if every vertex is consistent.

Node Consistency—II

A CSP can be made node consistent by applying the following procedure:

$$D_i \leftarrow D_i \cap \{x|P(x)\}$$

for each unary constraint P applied to x_i.
Equivalent to GFC on unary constraints.
Need only be applied once, no need for demons.

GFC in English—Unary Case

- Restrict the domain of x

GFC in English—Binary Case

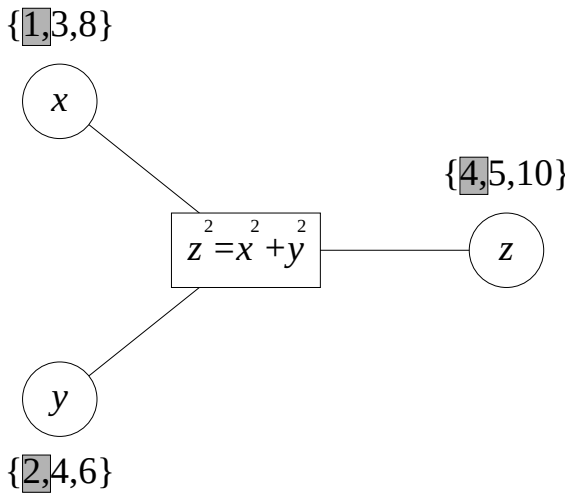
- Attach an after-demon to x that
  - Restricts the domain of y when x is bound
- Attach an after-demon to y that
  - Restricts the domain of x when y is bound

GFC in English—Ternary Case

- Attach an after-demon to x that
  - Restricts the domain of y when both x and z are bound
  - Restricts the domain of z when both x and y are bound
- Attach an after-demon to y that
  - Restricts the domain of z when both y and x are bound
  - Restricts the domain of x when both y and z are bound
- Attach an after-demon to z that
  - Restricts the domain of x when both z and y are bound
  - Restricts the domain of y when both z and x are bound

Arc Consistency—I

Relax the requirement that all but one variable be bound.



Arc Consistency—II

A hyperedge P between vertices $x_{i_1} \ldots x_{i_n}$ in a CSP hypergraph is consistent if:

$$
\begin{aligned}
&(\forall x_1 \in D_{i_1})(\exists x_2 \in D_{i_2}) \cdots (\exists x_n \in D_{i_n}) P(x_1, \ldots, x_n) \\
&\vdots \\
&(\forall x_k \in D_{i_k})(\exists x_1 \in D_{i_1}) \cdots (\exists x_{k-1} \in D_{i_{k-1}})(\exists x_{k+1} \in D_{i_{k+1}}) \cdots (\exists x_n \in D_{i_n}) \\
&\quad P(x_1, \ldots, x_n) \\
&\vdots \\
&(\forall x_n \in D_{i_n})(\exists x_1 \in D_{i_1}) \cdots (\exists x_{n-1} \in D_{i_{n-1}}) P(x_1, \ldots, x_n)
\end{aligned}
$$

A CSP is *arc consistent* if every hyperedge is consistent.

Arc Consistency—III

A CSP can be made arc consistent by applying the following procedure:

$$
\begin{aligned}
& D_{i_1} \leftarrow D_{i_1} \cap \{x_1 | (\exists x_2 \in D_{i_2}) \cdots (\exists x_n \in D_{i_n}) P(x_1, \ldots, x_n)\} \\
& \vdots \\
& D_{i_k} \leftarrow D_{i_k} \cap \left\{ x_k \;\middle|\; \begin{array}{l} (\exists x_1 \in D_{i_1}) \cdots (\exists x_{k-1} \in D_{i_{k-1}}) \\ \quad (\exists x_{k+1} \in D_{i_{k+1}}) \cdots (\exists x_n \in D_{i_n}) \\ \quad P(x_1, \ldots, x_n) \end{array} \right\} \\
& \vdots \\
& D_{i_n} \leftarrow D_{i_n} \cap \{x_n | (\exists x_1 \in D_{i_1}) \cdots (\exists x_{n-1} \in D_{i_{n-1}}) P(x_1, \ldots, x_n)\}
\end{aligned}
$$

to each hyperedge P.
Bounded by sum of domain sizes.
More constraint checks than Generalized Forward Checking.

AC in English—Unary Case

- Restrict the domain of x

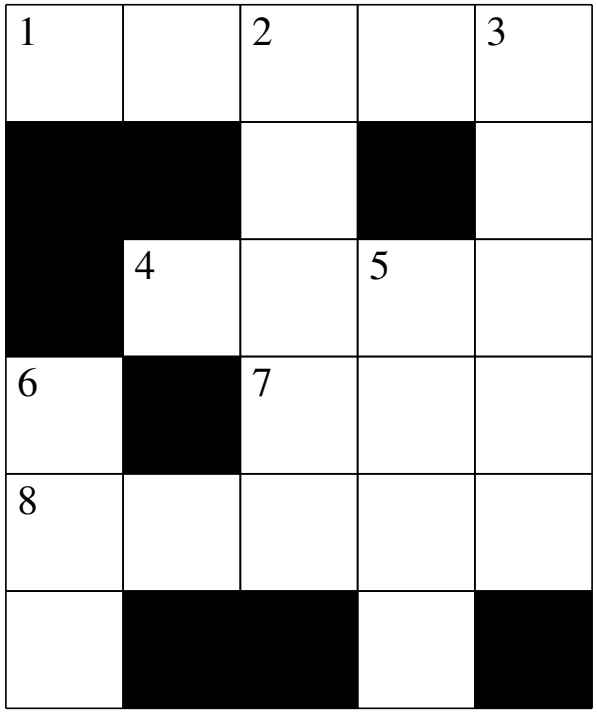
AC in English—Binary Case

- Attach an after-demon to x that
  - Restricts the domain of y
- Attach an after-demon to y that
  - Restricts the domain of x

AC in English—Ternary Case

- Attach an after-demon to x that
  - Restricts the domain of y
  - Restricts the domain of z
- Attach an after-demon to y that
  - Restricts the domain of z
  - Restricts the domain of x
- Attach an after-demon to z that
  - Restricts the domain of x
  - Restricts the domain of y

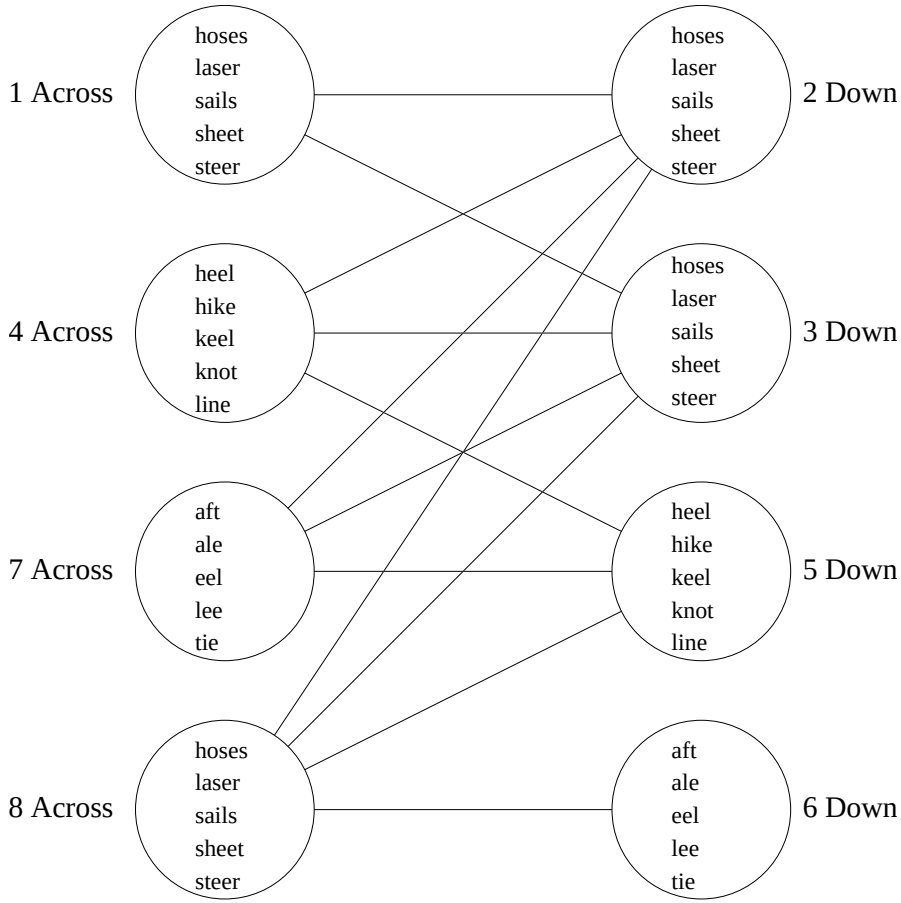
Solve Crossword Puzzles with Arc Consistency



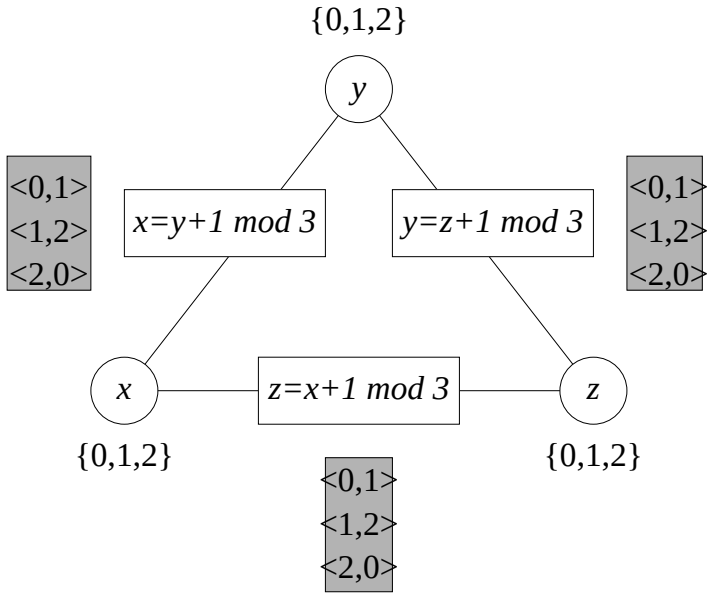
Word List

aft	ale	eel	hike	hoses	keel	knot
laser	lee	line	sails	sheet	steer	tie

CSP Graph for the Crossword Puzzle



Path Consistency—I



Two adjacent edges in a binary CSP graph are called a *path*.
A path in a binary CSP graph is consistent if:

$$(\forall x \in D_i)(\forall z \in D_k)[P_{ik}(x,z) \rightarrow (\exists y \in D_j)(P_{ij}(x,y) \wedge P_{jk}(y,z))]$$

Path Consistency—II

If P_{ij}, P_{jk}, or P_{ik} are unspecified, they are assumed to be the universally true predicate.
A binary CSP is *path consistent* if every path is consistent.
Making a binary CSP path consistent may require synthesizing new constraints.
Nonlocal but bounded.

(examines more than one constraint at a time)

GFC and AC as Inference Rules—I

Four types of propositions:

assignments: $x = a$

disassignments: $x \neq a$

set membership: $x \in \{a_1, \ldots, a_n\}$

conditional set membership: $x = a \rightarrow y \in \{b_1, \ldots, b_n\}$

GFC and AC as Inference Rules—II

Rule 1

$$\frac{\begin{array}{c} x \in \{a_1, \ldots, a_n\} \\ x \neq a_1 \\ \vdots \\ x \neq a_{i-1} \\ x \neq a_{i+1} \\ \vdots \\ x \neq a \end{array}}{x = a_i}$$

Rule 2

$$\frac{x = a}{x \neq b \quad \text{where } b \neq a}$$

Inference Rule for GFC

$$\frac{\begin{array}{c} x = a \rightarrow y \in \{b_1, \ldots, b_n\} \\ x = a \end{array}}{y \neq c \quad \text{where } c \notin \{b_1, \ldots, b_n\}}$$

Inference Rule for AC

$$\frac{\begin{array}{c} x = a \rightarrow y \in \{b_1, \ldots, b_n\} \\ y \neq b_1 \\ \vdots \\ y \neq b_n \end{array}}{x \neq a}$$

Issues in Consistency Techniques

- A CSP is inconsistent if any domain becomes empty.
- There is a single solution if all domains are singleton and network is node and arc consistent.
- Does path consistency imply arc consistency?
- Generalize to k-consistency (strong & weak).
- A CSP is consistent if it is strongly n-consistent.
  - Can be solved by GFC without backtracking.

Summary

- CSP is a general framework for expressing many search problems.
- Representing CSPs as directed (hyper)graphs.
- Relationship between CSP and SAT.
- Generate & Test induces search trees.
- Inference

early failure detection	node consistency
forward checking	arc consistency
value propagation	path consistency
generalized forward checking	k-consistency

- Applied
  - initially
  - incrementally, interleaved with backtracking search