Principles of Constraint Programming

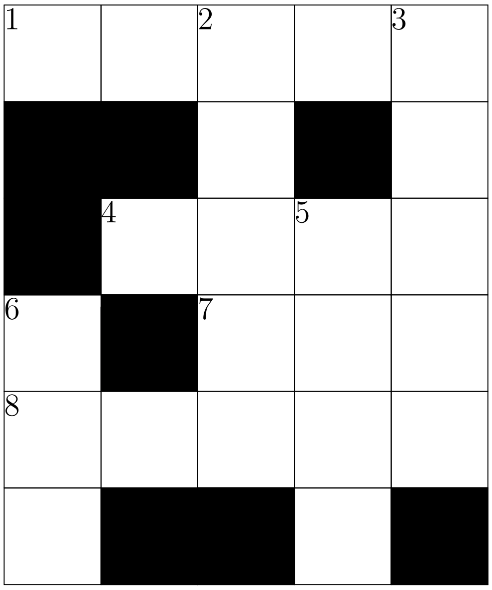
Krzysztof R. Apt

Chapter 7
Constraint Propagation Algorithms

Objectives

- Explain constraint propagation algorithms for the considered local consistency notions.
- Introduce generic iteration algorithms on partial orderings.
- Use them to explain constraint propagation algorithms.
- Discuss implementations of incomplete constraint solvers.

Motivation: Crossword Puzzle



- HOSES, LASER, SAILS, SHEET, STEER,
- HEEL, HIKE, KEEL, KNOT, LINE,
- AFT, ALE, EEL, LEE, TIE.

Variables: $x_1, \ldots, x_8$,
Domains: $x_7 \in \{\text{AFT, ALE, EEL, LEE, TIE}\}$, etc.
Constraints: one per crossing

$$\begin{aligned} C_{1,2} := \{&(\text{HOSES, SAILS}), (\text{HOSES, SHEET}),\\ &(\text{HOSES, STEER}), (\text{LASER, SAILS}),\\ &(\text{LASER, SHEET}), (\text{LASER, STEER})\}\ . \end{aligned}$$

etc.

Unique Solution

1 H	O	2 S	E	3 S
■	■	A	■	T
■	4 H	I	5 K	E
6 A	■	7 L	E	E
8 L	A	S	E	R
E	■	■	L	■

- We can reach it repeatedly applying *ARC CONSISTENCY* rules *1* and *2*.
- But many derivations exist.

General considerations:

- How to schedule rule applications to guarantee termination?
- How to avoid (at a low cost) redundant rule applications?
- Is the outcome of the derivations unique?
- If yes, how can it be characterised?

Informal Introduction

- **Constraint propagation** is a very widely used concept.

  On www.google.com on October 3, 2002:

"constraint propagation"	:	99300 hits,
"NP completeness"	:	92800 hits.

Other names:

- (local) consistency,
- consistency enforcing,
- Waltz,
- filtering,
- narrowing

algorithms

- (incomplete) constraint solver,
- reasoner.

Constraint Propagation: Intuition

Take a constraint satisfaction problem.

Repeatedly **reduce** its

- domains

and/or

- constraints

while maintaining equivalence.

Outcome: a **locally consistent** CSP.

Constraint Propagation Algorithms

- Scheduling of **atomic** reduction steps.
- Stopping criterion: **local consistency** notion.

Official History of Constraint Propagation

- Montanari '74: **path consistency** introduced and an algorithm to achieve it.
- Waltz '75: informally used in an algorithm for analysis of polyhedral scenes,
- Mackworth '77: **arc consistency** defined, arc consistency algorithm `AC-3` and path consistency algorithm `PC-2` introduced,
- ...
- Benhamou '96, Telerman and Ushakov '96: local consistency for domain reduction viewed as a greatest fixpoint,
- Van Emden '97 and Fages, Fowler, and Sola '98: constraint propagation linked to **chaotic** (i.e., fair) **iterations**,
- Apt '97, '99: arbitrary constraint propagation algorithms explained as **instances** of chaotic iteration algorithms.

Inofficial History of Constraint Propagation

Several techniques used in mathematics, operation research and computer science are instances of **constraint propagation**.

Examples

- various forms of resolution method (automated theorem proving),
- Gaussian elimination (linear algebra),
- Fourier-Motzkin elimination (linear programming),
- cutting planes (integer programming),
- various algorithms for test generation for digital circuits,
- ...

Approach

- **Constraint propagation** algorithms will be explained as special cases of **generic iteration algorithms**.
- We shall discuss these generic iterations algorithms first.
- Relevant properties of functions:
  - monotonicity,
  - inflationarity,
  - idempotence,
  - commutativity.
- We shall study such functions on partial orderings.
- Generic iterations algorithms schedule such functions.

Partial Orderings

A binary relation R on a set D is

- **reflexive** if $(a, a) \in R$ for all $a \in D$,
- **antisymmetric** if for all $a, b \in D$, $(a, b) \in R$ and $(b, a) \in R$ implies $a = b$,
- **transitive** if for all $a, b, c \in D$ $(a, b) \in R$ and $(b, c) \in R$ implies $(a, c) \in R$.

- **Partial ordering**: pair $(D, \sqsubseteq)$ with D a set and $\sqsubseteq$ a reflexive, antisymmetric and transitive relation on D.
- Given $(D, \sqsubseteq)$ an element d of D is the **least** element of D if $d \sqsubseteq e$ for all $e \in D$.

Examples

- $(\mathcal{N}, \leq)$,

  $\mathcal{N}$: the set of natural numbers,

- $(\mathcal{P}(A), \supseteq)$,

  $\mathcal{P}(A)$: the set of all subsets of the set A,

  $\supseteq$: the **reversed subset ordering**.

  Note: A is the least element of $\mathcal{P}(A)$.

- $(\mathcal{P}(D_1) \times \ldots \times \mathcal{P}(D_n), \supseteq)$

  $\mathcal{P}(D_1) \times \ldots \times \mathcal{P}(D_n)$:
  Cartesian product of $\mathcal{P}(D_i)$ for $i \in [1..n]$.

  $\supseteq$: the componentwise ordering $\supseteq$.

  So $(X_1, \ldots, X_n) \supseteq (Y_1, \ldots, Y_1)$ iff $X_i \supseteq Y_i$ for $i \in [1..n]$.

  Note: $(D_1, \ldots, D_n)$ is the least element of $\mathcal{P}(D_1) \times \ldots \times \mathcal{P}(D_n)$.

Fixpoints

Given: $(D, \sqsubseteq)$ and function f on D.

- a is a **fixpoint** of f if $f(a) = a$.
- a is the **least fixpoint** of f if a is the least element of the set $\{x \in D \mid f(x) = x\}$.

Iterations

Given: $(D, \sqsubseteq)$ with the least element $\perp$ and a set of functions $F := \{f_1, \ldots, f_k\}$ on D.

- **Iteration of F**: an infinite sequence of values $d_0, d_1, d_2, \ldots$ defined by

$$d_0 := \perp,$$

$$d_j := f_{i_j}(d_{j-1}),$$

where each $j > 0$ and $i_j \in [1..k]$.

- Increasing sequence $d_0 \sqsubseteq d_1 \sqsubseteq d_2 \ldots$ of elements from D **eventually stabilises at d** if for some $j \geq 0$

$$d_i = d \text{ for } i \geq j.$$

Stabilisation

Consider partial ordering $(D, \sqsubseteq)$ and functions f, g on D.

- f is **inflationary** if $x \sqsubseteq f(x)$.
- f is **monotonic** if $x \sqsubseteq y$ implies $f(x) \sqsubseteq f(y)$.
- f is **idempotent** if $ff(x) = f(x)$.
- f and g **commute** if $fg(x) = gf(x)$.
- f **semi-commutes with** g (*w.r.t.* $\sqsubseteq$) if $fg(x) \sqsubseteq gf(x)$.

Stabilisation Lemma

Given:

– $(D, \sqsubseteq)$ with the least element $\perp$,

– a finite set of monotonic functions F on D.

Suppose an iteration of F **eventually stabilises** at a common fixpoint d of functions from F. Then d is the **least** common fixed point of functions from F.

Commutativity

Given:

– $(D, \sqsubseteq)$ with the least element $\perp$,

– finite set $F := \{f_1, \ldots, f_k\}$ of functions on D s.t.

- each $f \in F$ is **monotonic** and **idempotent**,
- all $f, g \in F$ **commute**.

Then for each permutation $\pi : [1..k] \to [1..k]$

$$f_{\pi(1)} f_{\pi(2)} \cdots f_{\pi(k)}(\perp)$$

is the **least common fixpoint** of the functions from F.

DIRECT ITERATION algorithm (`DI`)

$d := \perp;$
$G := F;$
`WHILE` $G \neq \emptyset$ `DO`
   choose $g \in G$;
   $d := g(d)$;
   $G := G - \{g\}$
`END`

Semi-commutativity

Given: partial ordering $(D, \sqsubseteq)$ with the least element $\perp$ and a finite sequence $F := f_1, \ldots, f_k$ of

- **monotonic**,
- **inflationary** and
- **idempotent**

functions on D. Suppose

- f_i **semi-commutes** with f_j for $i > j$,

that is,

$$f_i f_j(x) \sqsubseteq f_j f_i(x) \text{ for all } x. \qquad (*)$$

Then $f_1 f_2 \ldots f_k(\perp)$ is the **least common fixpoint** of the functions from F.

(*) For example $f_3 f_1(x) \sqsubseteq f_1 f_3(x)$.

Simple Iteration Algorithm

SIMPLE ITERATION algorithm ($\texttt{SI}$)

```
d := ⊥;
FOR i := k TO 1 BY −1 DO
   d := f_i(d)
END
```

Note Upon termination $d = f_1 f_2 . . . f_k(\perp)$.

Theorem
Given: partial ordering $(D, \sqsubseteq)$ with the least element $\perp$ and a finite sequence $F := f_1, \ldots, f_k$ of

- **monotonic**,
- **inflationary** and
- **idempotent**

functions on D such that (*) holds. Then the $\texttt{SI}$ algorithm terminates and computes in d the **least common fixpoint** of functions from F.

Generic Iteration Algorithm (GI)

In absence of (semi-)commutativity information.

Given:

– $(D, \sqsubseteq)$ with the least element $\bot$,

– finite set $F := \{f_1, \ldots, f_k\}$ of functions on D.

$d := \bot;$
$G := F;$
WHILE $G \neq \emptyset$ DO
  choose $g \in G;$
  IF $d \neq g(d)$ THEN
    $G := G \cup update(G, g, d);$
    $d := g(d)$
  ELSE
    $G := G - \{g\}$
  END
END

where

$$\{f \in F - G \mid f(d) = d \wedge fg(d) \neq g(d)\} \subseteq update(G, g, d).$$

Properties of GI Algorithm

Generic Iteration Theorem

Consider **finite** partial ordering $(D, \sqsubseteq)$ with $\perp$ and functions $F := \{f_1, \ldots, f_k\}$ on D. Suppose all functions in F are

- **inflationary**,
- **monotonic**.

Then every execution of the `GI` algorithm

- **terminates**,
- computes in d the **least common fix-point** of the functions from F.

Instances of the Algorithms for Compound Domains

Suppose:

- $(D, \sqsubseteq)$ a Cartesian product of partial orderings,
- each function $f \in F$ defined on some Cartesian **subproduct**, determined by **scheme** (subsequence of $[1..n]$).
- For $f \in F$

  $$f^+ : D \to D.$$

  f^+ is the **canonic extension** of f.
- f and g **commute** if

  $$f^+ g^+(d) = g^+ f^+(d)$$

  for all $d \in D$.
- f **semi-commutes with** g (w.r.t. $\sqsubseteq$) if

  $$f^+ g^+(d) \sqsubseteq g^+ f^+(d)$$

  for all $d \in D$.

Instances of DI and SI Algorithms for Compound Domains

Direct Iteration algorithm

$d := (\perp_1, \ldots, \perp_n);$
$G := F_0;$
$\texttt{WHILE}\ G \neq \emptyset\ \texttt{DO}$
$\quad$ choose $g \in G$;
$\quad$ $d[s] := g(d[s])$, where s is the scheme of g
$\texttt{END}$

Simple Iteration algorithm

$d := (\perp_1, \ldots, \perp_n);$
$\texttt{FOR}\ i := k\ \texttt{TO}\ 1\ \texttt{BY}\ -1\ \texttt{DO}$
$\quad$ $d[s_i] := f_i(d[s_i])$, where s_i is the scheme of f_i
$\texttt{END}$

Instance of GI Algorithm for Compound Domains (CD)

Suppose:

- $(D, \sqsubseteq)$ a Cartesian product of partial orderings,
- each function $f \in F$ defined on some Cartesian subproduct, determined by scheme (subsequence of $[1..n]$).

$d := (\bot_1, \ldots, \bot_n)$;
$d' := d$;
$G := F_0$;
WHILE $G \neq \emptyset$ DO
  choose $g \in G$; suppose g is with scheme s;
  $d'[s] := g(d[s])$;
  IF $d'[s] \neq d[s]$ THEN
   $G := G \cup \{f \in F \mid f$ depends on i s.t. $d[i] \neq d'[i]\}$;
   $d[s] := d'[s]$
  ELSE
   $G := G - \{g\}$
  END
END

From Abstract Framework to Constraint Propagation

Consider a CSP

$$\langle C_1, \ldots, C_k \; ; \; x_1 \in D_1, \ldots, x_n \in D_n \rangle$$

- **Partial orderings with $\perp$:**
  - its elements:
    - for arc consistency: $(X_1, \ldots, X_n)$ such that $X_i \subseteq D_i$,
    - for path consistency: $(X_1, \ldots, X_k)$ such that $X_i \subseteq C_i$,
  - $\perp$:
    - for arc consistency: $(D_1, \ldots, D_n)$,
    - for path consistency: $(C_1, \ldots, C_k)$,
  - $\sqsubseteq$: componentwise reversed subset ordering $\supseteq$.
- **Inflationary and monotonic functions**: functions that reduce domains or constraints,
- **Common fixpoints**: correspond to CSP's that satisfy the considered notion of local consistency.

Node Consistency Algorithm

- CSP is **node consistent** if for every variable x every unary constraint on x coincides with the domain of x.

Node algorithm

$S_0 := \{C \mid C \text{ is a unary constraint from } \mathcal{C}\}$;
$S := S_0$;
$\texttt{WHILE}\ S \neq \emptyset\ \texttt{DO}$
  choose $C \in S$; suppose C is on x_i;
  $D_i := C \cap D_i$;
  $S := S - \{C\}$
$\texttt{END}$

- An instance of the Direct Iteration algorithm for compound domains.
- It can be systematically derived from it by choosing the appropriate partial ordering and functions.

Arc Consistency: Recap

- A constraint C on the variables x, y with the domains X and Y (so $C \subseteq X \times Y$) is **arc consistent** if
  - $\forall a \in X \exists b \in Y \; (a, b) \in C$,
  - $\forall b \in Y \exists a \in X \; (a, b) \in C$.
- A CSP is **arc consistent** if all its binary constraints are.

Arc Consistency: Recap

$$\textit{ARC CONSISTENCY 1}$$

$$\frac{\langle C \ ; \ x \in D_x, y \in D_y \rangle}{\langle C \ ; \ x \in D'_x, y \in D_y \rangle}$$

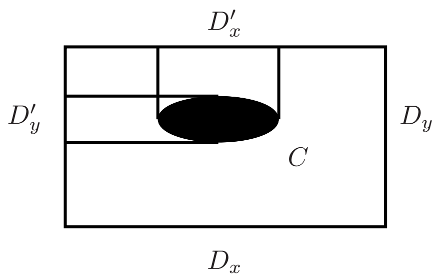
where $D'_x := \{a \in D_x \mid \exists\, b \in D_y \ (a, b) \in C\}$

$$\textit{ARC CONSISTENCY 2}$$

$$\frac{\langle C \ ; \ x \in D_x, y \in D_y \rangle}{\langle C \ ; \ x \in D_x, y \in D'_y \rangle}$$

where $D'_y := \{b \in D_y \mid \exists\, a \in D_x \ (a, b) \in C\}$.

Intuition



Note A CSP is arc consistent iff it is closed under the applications of the *ARC CONSISTENCY* rules 1 and 2.

Projection Functions

Given: $C \subseteq X \times Y$.

Let

$$X' = \{a \in X \mid \exists b \in Y \ (a, b) \in C\},$$
$$Y' = \{b \in Y \mid \exists a \in X \ (a, b) \in C\}.$$

Define

$$\pi_1(X, Y) := (X', Y),$$
$$\pi_2(X, Y) := (X, Y').$$

ARC CONSISTENCY rule *1* orresponds to function π_1 on $\mathcal{P}(D_x) \times \mathcal{P}(D_y)$.

ARC CONSISTENCY rule *2* orresponds to function π_2 on $\mathcal{P}(D_x) \times \mathcal{P}(D_y)$.

Arc Consistency as a Fixpoint

π_i^+: canonic extension of π_i to all domains.

Lemma

- $\langle \mathcal{C} \; ; \; x_1 \in D_1, \ldots, x_n \in D_n \rangle$ is arc consistent iff $(D_1, \ldots, D_n)$ is a **common fixpoint** of all functions π_1^+ and π_2^+.
- Each projection function π_i is
  - **inflationary** w.r.t. the componentwise ordering $\supseteq$,
  - **monotonic** w.r.t. the componentwise ordering $\supseteq$.

Conclusion:

- We can instantiate the CD algorithm with the projection functions.
- Call it `ARC` algorithm.

ARC ALGORITHM

$S_0 := \{C \mid C \text{ is a binary constraint from } \mathcal{C}\} \cup$
$\qquad \{C^T \mid C \text{ is a binary constraint from } \mathcal{C}\};$
$S := S_0;$
`WHILE` $S \neq \emptyset$ `DO`
  choose $C \in S$; suppose C is on x_i, x_j;
  $D_i := \{a \in D_i \mid \exists\, b \in D_j\ (a, b) \in C\};$
  `IF` D_i changed `THEN`
    $S := S \cup \{C' \in S_0 \mid C' \text{ is on } y, z \text{ where } y \text{ is } x_i \text{ or } z \text{ is } x_i\}$
  `ELSE`
    $S := S - \{C\}$
  `END`
`END`

Theorem

Consider $\mathcal{P} := \langle \mathcal{C}\ ;\ x_1 \in D_1, \ldots, x_n \in D_n \rangle$ where each D_i is finite.

The `ARC` algorithm always terminates. Let $\mathcal{P}'$ be the CSP determined by $\mathcal{P}$ and the sequence of the computed domains. Then

- $\mathcal{P}'$ is arc consistent,
- $\mathcal{P}'$ is equivalent to $\mathcal{P}$.

Hyper-arc Consistency: Recap

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

$$\forall i \in [1..n] \forall a \in D_i \; \exists d \in C \; a = d[x_i].$$

- CSP is **hyper-arc consistent** if all its constraints are.

$$\textit{HYPER-ARC CONSISTENCY}$$

$$\frac{\langle C \; ; \; x_1 \in D_1, \ldots, x_k \in D_k \rangle}{\langle C \; ; \; \ldots, x_i \in D'_i, \ldots \rangle}$$

C a constraint on the variables $x_1, \ldots, x_k$, $i \in [1..k]$,

$$D'_i := \{a \in D_i \mid \exists d \in C \; a = d[x_i]\}.$$

Note A CSP is hyper-arc consistent iff it is closed under the applications of the *HYPER-ARC CONSISTENCY* rule.

Hyper-arc Consistency a Fixpoint

C: a constraint on $x_1, \ldots, x_k$ with respective domains $D_1, \ldots, D_k$.

For each $i \in [1..k]$
HYPER-ARC CONSISTENCY rule corresponds to function π_i on $\mathcal{P}(D_1) \times \cdots \times \mathcal{P}(D_k)$:

$$\pi_i(X_1, \ldots, X_k) := (X_1, \ldots, X_{i-1}, X_i', X_{i+1}, \ldots, X_k)$$

where

$$X_i' = \{d[i] \mid d \in X_1 \times \cdots \times X_k \text{ and } d \in C\}.$$

Each π_i is associated with a constraint C.

Theorem

- A CSP $\langle \mathcal{C} \; ; \; x_1 \in D_1, \ldots, x_n \in D_n \rangle$ is hyper-arc consistent iff $(D_1, \ldots, D_n)$ is a common fixpoint of all functions π_i^+.
- Each function π_i is
  - inflationary w.r.t. the componentwise ordering $\supseteq$,
  - monotonic w.r.t. the componentwise ordering $\supseteq$.

Hyper-arc Consistency Algorithm

Instantiate the `CD` algorithm with

$$F_0 := \{f \mid f \text{ is a } \pi_i \text{ function associated with a constraint of } \mathcal{P}\}$$

and each $\perp_i := D_i$.

Call it `HYPER-ARC` algorithm.

Theorem
Consider $\mathcal{P} := \langle \mathcal{C} \ ; \ x_1 \in D_1, \ldots, x_n \in D_n \rangle$ where each D_i is finite.

The `HYPER-ARC` algorithm always terminates. Let $\mathcal{P}'$ be the CSP determined by $\mathcal{P}$ and the sequence of the domains computed in d. Then

- $\mathcal{P}'$ is hyper-arc consistent,
- $\mathcal{P}'$ is equivalent to $\mathcal{P}$.

Path Consistency: Recap

A normalised CSP is **path consistent** if for each subset $\{x, y, z\}$ of its variables

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

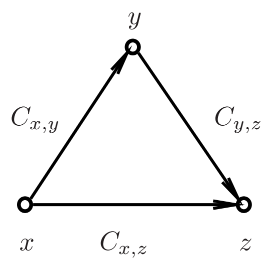
Note A normalised CSP is path consistent iff for each subsequence x, y, z of its variables

$$C_{x,y} \subseteq C_{x,z} \cdot C^T_{y,z},$$

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

$$C_{y,z} \subseteq C^T_{x,y} \cdot C_{x,z}.$$

Intuition



Path Consistency: Recap

PATH CONSISTENCY 1

$$\frac{C_{x,y}, C_{x,z}, C_{y,z}}{C'_{x,y}, C_{x,z}, C_{y,z}}$$

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

PATH CONSISTENCY 2

$$\frac{C_{x,y}, C_{x,z}, C_{y,z}}{C_{x,y}, C'_{x,z}, C_{y,z}}$$

where $C'_{x,z} := C_{x,z} \cap C_{x,y} \cdot C_{y,z}$,

PATH CONSISTENCY 3

$$\frac{C_{x,y}, C_{x,z}, C_{y,z}}{C_{x,y}, C_{x,z}, C'_{y,z}}$$

where $C'_{y,z} := C_{y,z} \cap C^T_{x,y} \cdot C_{x,z}$.

Note A normalised CSP is path consistent iff it is closed under the applications of the *PATH CONSISTENCY* rules 1, 2 and 3.

$f^z_{x,y}$ functions

$\mathcal{P}$: a normalised CSP with binary constraints $C_1, \ldots, C_k$.

PATH CONSISTENCY rule *1* corresponds to

$$f^z_{x,y}(P, Q, R) := (P', Q, R),$$

where

$$P' := P \cap Q \cdot R^T,$$

PATH CONSISTENCY rule *2* corresponds to

$$f^y_{x,z}(P, Q, R) := (P, Q', R),$$

where

$$Q' := Q \cap P \cdot R,$$

PATH CONSISTENCY rule *3* corresponds to

$$f^x_{y,z}(P, Q, R) := (P, Q, R'),$$

where

$$R' := R \cap P^T \cdot Q.$$

Path Consistency as a Fixpoint

$(f^z_{x,y})^+$, $(f^y_{x,z})^+$ and $(f^x_{y,z})^+$ are functions on

$$\mathcal{P}(C_1) \times \cdots \times \mathcal{P}(C_k).$$

Theorem

- A standardised CSP $\mathcal{P}$ with the binary constraints $C_1, \ldots, C_k$ is path consistent iff $(C_1, \ldots, C_k)$ is a common fixpoint of all functions $(f^z_{x,y})^+$, $(f^y_{x,z})^+$ and $(f^x_{y,z})^+$ associated with the subsequences x, y, z of vars of $\mathcal{P}$.
- The functions $f^z_{x,y}$, $f^y_{x,z}$ and $f^x_{y,z}$ are
  - inflationary w.r.t. the componentwise ordering $\supseteq$,
  - monotonic w.r.t. the componentwise ordering $\supseteq$.

A Path Consistency Algorithm

Instantiate the CD algorithm with

$$F_0 := \{f^z_{x,y}, f^y_{x,z}, f^x_{y,z} \mid x, y, z \text{ is a subsequence } quad \text{ of variables of } \mathcal{P}\},$$

$n := k$ and each $\perp_i := C_i$.

Call it PATH algorithm.

Theorem $\mathcal{P}$ a standardised CSP with the finite binary constraints $C_1, \ldots, C_k$.

The PATH algorithm always terminates. Let $\mathcal{P}'$ be the CSP obtained from $\mathcal{P}$ by replacing its binary constraints by the binary constraints computed in d. Then

- $\mathcal{P}'$ is path consistent,
- $\mathcal{P}'$ is equivalent to $\mathcal{P}$.

Directional Arc Consistency as a Fixpoint

Darc Theorem Consider a CSP $\mathcal{P}$ and linear ordering $\prec$ on its variables. Let

$$\mathcal{P}_{\prec} := \langle \mathcal{C} \ ; \ x_1 \in D_1, \ldots, x_n \in D_n \rangle.$$

where $x_1 \prec x_2 \ldots \prec x_n$.

Then

- $\mathcal{P}$ is directionally arc consistent w.r.t. $\prec$ iff $(D_1, \ldots, D_n)$ is a **common fixpoint** of all functions π_1^+ associated with the binary constraints from $\mathcal{P}_{\prec}$.
- Each projection function π_i is idempotent.
- Consider two binary constraints of $\mathcal{P}_{\prec}$, C_1 on $\_, z$ and C_2 on $\_, y$, where $y \preceq z$.

  Then the π_1 function of C_1 semi-commutes with the π_1 function of C_2 w.r.t. $\supseteq$.

Ordering of π_1 Functions

Assume $\mathcal{P}$ is standardised.
Suppose

$$\mathcal{P}_{\prec} := \langle \mathcal{C} \ ; \ x_1 \in D_1, \ldots, x_n \in D_n \rangle.$$

So $x_1 \prec x_2 \prec \ldots \prec x_n$.

Denote the constraint of $\mathcal{P}_{\prec}$ on x_i, x_j by $C_{i,j}$.
Order the binary constraints of $\mathcal{P}_{\prec}$ as follows:

$$\begin{array}{llll} C_{1,n}, & C_{2,n}, & \ldots, C_{n-2,n}, & C_{n-1,n}, \\ C_{1,n-1}, & C_{2,n-1}, & \ldots, C_{n-2,n-1}, & \\ & \ldots & & \\ C_{1,2}. & & & \end{array}$$

For the corresponding sequence of the π_1 functions Darc Theorem applies.

Instantiate the `SICD` algorithm with

- the above sequence of the π_1 functions,
- each $\bot_i := D_i$.

Call it Directional Arc Consistency Algorithm (`DARC`).

DARC Algorithm

```
FOR j := n TO 2 BY −1 DO
  FOR i := 1 TO j − 1 DO
    D_i := {a ∈ D_i | ∃ b ∈ D_j (a, b) ∈ C_{i,j}}
  END
END
```

Theorem Consider a CSP $\mathcal{P}$ with a linear ordering $\prec$ on its variables. Let

$$\mathcal{P}_{\prec} := \langle \mathcal{C} \; ; \; x_1 \in D_1, \ldots, x_n \in D_n \rangle.$$

The DARC algorithm always terminates. Let $\mathcal{P}'$ be the CSP determined by $\mathcal{P}_{\prec}$ and the sequence of the domains computed in d. Then

- $\mathcal{P}'$ is directionally arc consistent w.r.t. $\prec$,
- $\mathcal{P}'$ is equivalent to $\mathcal{P}_{\prec}$.

Implementations of Incomplete Constraint Solvers

Lemma Consider a domain reduction rule R.

Suppose the domains in conclusion of R are built from the domains in premise of R using these operations on relations:

- union and intersection,
- transposition operation "$.^T$",
- composition operation "$. \cdot ..$",
- join operation $\bowtie$,
- projection functions π_i and Π_X, and
- removal of an element.

Then R viewed as function on the variable domains is inflationary and monotonic w.r.t. the componentwise ordering $\supseteq$.

Conclusion

- We can instantiate the generic iteration algorithm `GI` by such domain reduction rules.
- This yields implementations of incomplete constraint solvers of Chapter 6.

Other Local Consistency Notions

This approach applies to other local consistency notions. Generic iteration algorithm `GI` can be used to derive constraint propagation algorithms for

- directional path consistency,
- k-consistency,
- strong k-consistency,
- relational consistency.

Objectives

- Explain constraint propagation algorithms for the considered local consistency notions.
- Introduce generic iteration algorithms on partial orderings.
- Use them to explain constraint propagation algorithms.
- Discuss implementations of incomplete constraint solvers.