Lecture 6

Constraint Propagation

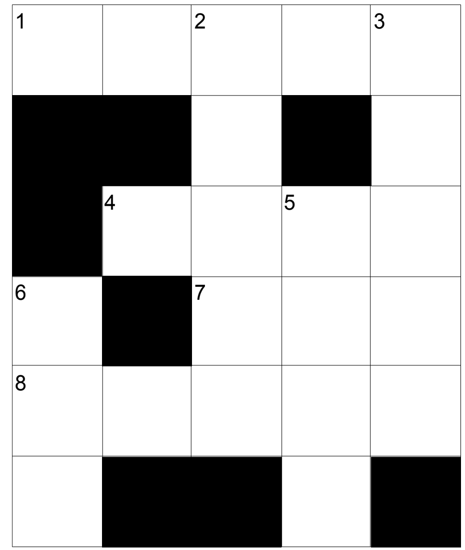
Outline

- Explain constraint propagation algorithms for various 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

Fill the crossword grid with words from

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



Variables: $x_1, \ldots, x_8$

Domains: $x_7 \in$ {AFT, ALE, EEL, LEE, TIE}, etc.

Constraints: one per crossing

$C_{1,2}$:= {(HOSES, SAILS), (HOSES, SHEET), (HOSES, STEER), (LASER, SAILS), (LASER, SHEET), (LASER, STEER)}

etc.

Unique Solution

- We can solve it by 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 low cost) redundant rule applications?
- Is the outcome of the derivations unique?
- If so, how can it be characterized?

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	

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

Approach

- Constraint propagation algorithms will be explained as special cases of generic iteration algorithms
- We shall discuss these generic iteration algorithms first
- Relevant properties of functions:
  - monotonicity
  - inflationarity
  - idempotence
  - commutativity
- We shall study such functions on partial orderings
- Generic iteration 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 $d \in D$ is the least element of D if $d \sqsubseteq e$ for all $e \in D$

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 $F \coloneqq \{f_1, ..., f_k\}$ on D

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

  $$d_0 \coloneqq \perp$$

  $$d_j \coloneqq f_{i_j}(d_{j-1})$$

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

- Increasing sequence $d_0 \sqsubseteq d_1 \sqsubseteq d_2 ...$ of elements from D eventually stabilizes 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 $f(f(x)) = f(x)$
- f and g commute if $f(g(x)) = g(f(x))$
- f semi-commutes with g (w.r.t. $\sqsubseteq$) if $f(g(x)) \sqsubseteq g(f(x))$

Lemma

Given:

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

- a finite set of monotonic functions F on D

Suppose an iteration of F eventually stabilizes at a common fixpoint d of functions from F. Then d is the least common fixpoint of functions from F.

Commutativity

Given:

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

- finite set $F \coloneqq \{f_1, ..., f_k\}$ of functions on D such that

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

Then for each permutation π: $[1..k] \rightarrow [1..k]$

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

is the least common fixpoint of the functions from F.

Direct Iteration Algorithm

procedure DIRECT ITERATION

  $d \coloneqq \bot$;

  $G \coloneqq F$;

  while $G \neq \emptyset$ **do**

    choose $g \in G$;

    $d \coloneqq g(d)$

    $G \coloneqq G - \{g\}$

  end-while

end

Semi-Commutativity

Given:

- partial ordering $(D, \sqsubseteq)$ with the least element $\perp$

- finite sequence $F \coloneqq f_1, \ldots, f_k$ of

- monotonic
- inflationary and
- idempotent

functions on D.

Suppose f_i semi-commutes with f_j for $i > j$.

Then

$$f_1 f_2 \ldots f_k(\perp)$$

is the least common fixpoint of the functions from F.

Simple Iteration Algorithm

procedure SIMPLE ITERATION

$d \coloneqq \perp;$

for $i \coloneqq k$ **to** 1 **by** – 1 **do**

$d \coloneqq f_i(d)$

end-for

end

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

Theorem

Given: partial ordering $(D, \sqsubseteq)$ with the least element $\perp$ and a finite sequence $F \coloneqq f_1, \ldots, f_k$ of monotonic, inflationary, and idempotent functions on D such that f_i semi-commutes with f_j for $j < i$. Then the algorithm terminates and computes in d the least common fixpoint of functions from F.

Generic Iteration Algorithm

In the absence of (semi-)commutativity information

Given: - $(D, \sqsubseteq)$ with the least element $\perp$

- finite set $F \coloneqq \{f_1, ..., f_k\}$ of functions on D

procedure GENERIC ITERATION

$d \coloneqq \perp$;

$G \coloneqq F$;

while $G \neq \emptyset$ **do**

choose $g \in G$;

if $d \neq g(d)$ **then** $G \coloneqq G \cup update(G, g, d)$; $d \coloneqq g(d)$

else $G \coloneqq G - \{g\}$

end-while

end

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

Properties of GI Algorithm

Theorem

Consider finite partial ordering $(D, \sqsubseteq)$ with $\perp$ and functions $F \coloneqq \{f_1, ..., f_k\}$ on D.

Suppose all functions in F are inflationary and monotonic.

Then every execution of the GI algorithm terminates and computes in d the least common fixpoint of the functions from F.

Instances for Compound Domains

Suppose:

- $(D, \sqsubseteq)$ a Cartesian product of partial orderings
- each function $f \in F_0$ defined on some Cartesian subproduct, determined by scheme (subsequence of $[1..n]$)
- For $f \in F_0$
$$f^+ : D \to D$$
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 for Compound Domains, ctd

procedure DIRECT ITERATION

$d \coloneqq (\perp_1, ..., \perp_n)$;

$G \coloneqq F_0$;

while $G \neq \emptyset$ **do**

choose $g \in G$; $G \coloneqq G - \{g\}$;

$d[s] \coloneqq g(d[s])$ where s is the scheme of g

end-while

end

procedure SIMPLE ITERATION

$d \coloneqq (\perp_1, ..., \perp_n)$;

for $i \coloneqq k$ **to** 1 **by** – 1 **do** where s_i is the scheme of f_i

$d[s_i] \coloneqq f_i(d[s_i])$

end

Instances for Compound Domains, ctd

Suppose:

- (D, $\sqsubseteq$) a Cartesian product of partial orderings
- each function $f \in F_0$ defined on some Cartesian subproduct, determined by scheme (subsequence of [1..n])

procedure COMPUND DOMAIN

$d, d' \coloneqq (\perp_1, ..., \perp_n)$;

$G \coloneqq F_0$;

while $G \neq \emptyset$ **do**

choose $g \in G$; suppose g has scheme s;

$d'[s] \coloneqq g(d[s])$;

if $d'[s] \neq d[s]$ **then** $G \coloneqq \cup \{f \in F \mid f$ depends on an i in s such that $d[i] \neq d'[i]\}$;

$d[s] \coloneqq d'[s]$

else $G \coloneqq G - \{g\}$

end-while

end

From Abstract Framework to Constraint Propagation

Consider a CSP $\langle C_1, ..., C_k ; x_1 \in D_1, ..., x_n \in D_n\rangle$

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

  $S_0 \coloneqq \{C \mid C$ is a unary constraint from $\mathcal{C}\}$;
  $S \coloneqq S_0$;
  while $S \neq \emptyset$ **do**
      choose $C \in S$;   suppose C is on x_i;
      $D_i \coloneqq C \cap D_i$;
      $S \coloneqq S - \{C\}$
  end-while

- 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

ARC CONSISTENCY 1

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

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

ARC CONSISTENCY 2

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

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

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 corresponds to function π_1 on $\mathcal{P}(D_x) \times \mathcal{P}(D_y)$

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

Arc Consistency as Fixpoint

π_i^+: canonic extension of π_i to all domains in the CSP

Lemma

- $\langle \mathcal{C} ; x_1 \in D_1, ..., x_n \in D_n \rangle$ is arc consistent iff $(D_1, ..., D_n)$ 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 COMPOUND DOMAIN algorithm (cf. Slide 22) with the projection functions
- Call it ARC algorithm

ARC Algorithm

procedure ARC

$S_0 \coloneqq \{C \mid C \text{ is a binary constraint from } \mathcal{C}\} \cup$

$\{C^T \mid C \text{ is a binary constraint from } \mathcal{C}\}$;

$S \coloneqq S_0$;

while $S \neq \emptyset$ **do**

choose $C \in S$; suppose C is on x_i, x_j;

$D_i \coloneqq \{a \in D_i \mid \exists b \in D_j\ (a, b) \in C\}$;

if D_i changed **then**

$S \coloneqq S \cup \{C' \in S_0 \mid C' \text{ is on } y, z \text{ where } y \text{ is } x_i \text{ or } z \text{ is } x_i$

else $S \coloneqq S - \{C\}$

end-while

end

Properties of ARC Algorithm

Theorem

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

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 $x_1, \ldots, x_k$, $i \in [1..k]$, $D'_i := \{a \in D_i \mid \exists d \in C \; a = d[x_i]\}$

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

Hyper-Arc Consistency as Fixpoint

C: a constraint on $x_1, ..., x_k$ with respective domains $D_1, ..., D_k$. For each $i \in [1..k]$ HYPER-ARC CONSISTENCY rule corresponds to function π_i on $\mathcal{P}(D_1) \times \cdots \times \mathcal{P}(D_k)$:

$$\pi_i\,(X_1, ..., X_k) \coloneqq (X_1, ..., X_{i-1}, X'_i, X_{i+1}, ..., X_k)$$

where $X'_i = \{d[x_i] \mid d \in X_1 \times \cdots \times X_k \text{ and } d \in C\}$

Each π_i is associated with a constraint C

Theorem

- A CSP $\langle \mathcal{C}\ ; x_1 \in D_1, ..., x_n \in D_n \rangle$ is hyper-arc consistent iff $(D_1, ..., D_n)$ 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 COMPOUND DOMAIN algorithm (cf. Slide 22) with

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

and each $\perp_i \coloneqq D_i$

Call it `HYPER-ARC` algorithm

Theorem

Consider $\mathcal{P} \coloneqq \langle \mathcal{C} ; x_1 \in D_1, ..., x_n \in D_n \rangle$ 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 $\mathcal{P}$

Implementation 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 $\prod_X$
- 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 by such domain reduction rules. This yields implementations of incomplete constraint solvers of Chapter 5.

Other Local Consistency Notions

This approach applies to other local consistency notions.
The GENERIC ITERATION algorithm 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 various local consistency notions
- Introduce generic interation algorithms on partial orderings
- Use them to explain constraint propagation algorithms
- Discuss implementations of incomplete constraint solvers