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

Constraints: one per crossing

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

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?

![Crossword Puzzle](Image)
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 \(F := \{f_1, \ldots, f_k\}\) 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 \(j > 0\) and each \(i_j \in [1..k]\)

- Increasing sequence \(d_0 \sqsubseteq d_1 \sqsubseteq d_2 \ldots\) 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 \(\bot\)

- 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:
- $(D, \subseteq)$ with the least element $\bot$
- finite set $F := \{f_1, \ldots, f_k\}$ of functions on $D$ such that
  - each $f \in F$ is monotonic and idempotent
  - all $f, g \in F$ commute

Then for each permutation $\pi: [1..k] \rightarrow [1..k]$
$$f_{\pi(1)}f_{\pi(2)}\cdots f_{\pi(k)}(\bot)$$
is the least common fixpoint of the functions from $F$. 
Direct Iteration Algorithm

procedure DIRECT ITERATION
    \[ d := \bot; \]
    \[ G := F; \]
    \[ \textbf{while } G \neq \emptyset \textbf{ do} \]
    \[ \text{choose } g \in G; \]
    \[ d := g(d) \]
    \[ G := G \setminus \{g\} \]
    \[ \textbf{end-while} \]
end
Semi-Commutativity

Given:
- partial ordering \((D, \sqsubseteq)\) with the least element \(\bot\)
- 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 \(i > j\).
Then
\[
f_1 f_2 \ldots f_k(\bot)
\]
is the least common fixpoint of the functions from \(F\).
Simple Iteration Algorithm

procedure SIMPLE ITERATION
    \[d := \bot;\]
    \[\text{for } i := k \text{ to } 1 \text{ by } -1 \text{ do}\]
    \[d := f_i(d)\]
    \[\text{end-for}\]
end

Note: Upon termination \(d = f_1f_2 \ldots f_k(\bot)\)

Theorem
Given: partial ordering \((D, \sqsubseteq)\) with the least element \(\bot\) and a finite sequence \(F := f_1, \ldots, f_k\) 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: - \((D, \subseteq)\) with the least element \(\bot\)
- finite set \(F := \{f_1, \ldots, f_k\}\) of functions on \(D\)

procedure GENERIC ITERATION

\[
d := \bot;
G := F;
\]

\[\text{while } G \neq \emptyset \text{ do}
\]

choose \(g \in G\);

\[
\text{if } d \neq g(d) \text{ then } G := G \cup \text{update}(G, g, d); d := g(d)
\]

\[\text{else } G := G - \{g\}\]

end-while

end

where \(\{f \in F - G \mid f(d) = d \land f(g(d)) \neq g(d)\} \subseteq \text{update}(G, g, d)\)
Properties of GI Algorithm

**Theorem**
Consider finite partial ordering \((D, \sqsubseteq)\) with \(\bot\) and functions \(F := \{f_1, \ldots, f_k\}\) 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:

- \((D, \subseteq)\) 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 \(f \in F_0\)
  \[ f^+ : D \rightarrow D \]
  \(f^+\) 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. \(\subseteq\)) if
  \[ f^+(g^+(d)) \subseteq g^+(f^+(d)) \]
  for all \(d \in D\)
Instances for Compound Domains, ctd

**procedure** DIRECT ITERATION

\[ d := (\perp_1, \ldots, \perp_n); \]
\[ G := F_0; \]

**while** \( G \neq \emptyset \) **do**

\[ \text{choose } g \in G; \quad G := G - \{g\}; \]

\[ d[s] := g(d[s]) \quad \text{where } s \text{ is the scheme of } g \]

**end-while**

**end**

**procedure** SIMPLE ITERATION

\[ d := (\perp_1, \ldots, \perp_n); \]

\[ \text{for } i := k \text{ to } 1 \text{ by } -1 \text{ do} \quad \text{where } s_i \text{ is the scheme of } f_i \]

\[ d[s_i] := f_i(d[s_i]) \]

**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 \([1..n]\))

```plaintext
procedure COMPUND DOMAIN
  \(d, d' \leftarrow (\bot_1, \ldots, \bot_n)\);
  \(G \leftarrow F_0\);
  while \(G \neq \emptyset\) do
    choose \(g \in G\);
    suppose \(g\) has scheme \(s\);
    \(d'[s] \leftarrow g(d[s])\);
    if \(d'[s] \neq d[s]\) then
      \(G \leftarrow \cup \{f \in F \mid f\) depends on an \(i\) in \(s\) such that \(d[i] \neq d'[i]\}\);
      \(d[s] \leftarrow d'[s]\)
    else
      \(G \leftarrow G - \{g\}\)
  end-while
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
  - 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 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 $x$

$$S_0 := \{ C \mid C \text{ is a unary constraint from } C \};$$
$$S := S_0;$$
**while** $S \neq \emptyset$ **do**
  choose $C \in S; \quad \text{suppose } C \text{ is on } x_i;$$
  $$D_i := C \cap D_i;$$
  $$S := 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 consistent.
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 $\pi_1$ on $\mathcal{P}(D_x) \times \mathcal{P}(D_y)$

ARC CONSISTENCY rule 2 corresponds to function $\pi_2$ on $\mathcal{P}(D_x) \times \mathcal{P}(D_y)$
Arc Consistency as Fixpoint

\( \pi_i^+ \): canonic extension of \( \pi_i \) to all domains in the CSP

**Lemma**

- \( \langle C ; x_1 \in D_1, ..., x_n \in D_n \rangle \) is arc consistent iff \( (D_1, ..., D_n) \) is a common fixpoint of all functions \( \pi_1^+ \) and \( \pi_2^+ \)
- Each projection function \( \pi_i \) 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 \( \text{ARC} \) algorithm
ARC Algorithm

procedure ARC

\[ S_0 := \{ C \mid C \text{ is a binary constraint from } \mathcal{C} \} \cup \{ C^T \mid C \text{ is a binary constraint from } \mathcal{C} \}; \]
\[ S := S_0; \]
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-while

end
Properties of \textit{ARC} Algorithm

\textbf{Theorem}

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

The \textit{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

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

  $C$ 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 \(x_1, \ldots, x_k\) with respective domains \(D_1, \ldots, D_k\). For each \(i \in [1..k]\)

HYPER-ARC CONSISTENCY rule corresponds to function \(\pi_i\) 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 \(X'_i = \{d[x_i] \mid d \in X_1 \times \cdots \times X_k\ \text{and} \ d \in C\}\)

Each \(\pi_i\) is associated with a constraint \(C\)

**Theorem**

- A CSP \(\langle C ; x_1 \in D_1, \ldots, x_n \in D_n \rangle\) is hyper-arc consistent iff \((D_1, \ldots, D_n)\) is a common fixpoint of all functions \(\pi_i^+\)
- Each function \(\pi_i\) 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
\[ F_0 := \{ f \mid f \text{ is a } \pi_i \text{ function associated with a constraint of } \mathcal{P} \} \]
and each \( \perp_i := D_i \)
Call it \textsc{Hyper-Arc algorithm}

**Theorem**
Consider \( \mathcal{P} := \langle C ; x_1 \in D_1, \ldots, x_n \in D_n \rangle \) where each \( D_i \) is finite.
The \textsc{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 “.⋅..”
- join operation $\Join$
- projection functions $\pi_i$ and $\Pi_X$
- 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 iteration algorithms on partial orderings
- Use them to explain constraint propagation algorithms
- Discuss implementations of incomplete constraint solvers