Introduction to Programming

Bertrand Meyer

Last revised 23 January 2004
Lecture 24: Topological Sort — 1: Background
Un dîner en famille.

— Surtout ! ne parlons pas de l’affaire Dreyfus!
“Topological sort”

From a given partial order, produce a compatible total order
Un dîner en famille.

– Surtout ! ne parlons pas de l’affaire Dreyfus!
Un dîner en famille.

– Surtout ! ne parlons pas de l’affaire Dreyfus !

... Ils en ont parlé ...

Chair of Software Engineering

Introduction to Programming – Lecture 24

by Caran d’Ache
The problem

From a given partial order, produce a compatible total order

- **Partial order**: ordering constraints between elements of a set, e.g.
  - “Remove dishes *before* discussing politics”
  - “Walk to Ütliberg *before* lunch”
  - “Take your medicine *before* lunch”
  - “Finish lunch *before* removing dishes”

- **Total order**: sequence including all elements of set

- **Compatible**: the sequence respects all ordering constraints
  - Ütliberg, Medicine, Lunch, Remove, Discuss : OK
  - Medicine, Ütliberg, Lunch, Remove, Discuss : OK
  - Discuss, Medicine, Lunch, Remove, Ütliberg : not OK
- “Remove dishes \textit{before} discussing politics”
- “Walk to Útliberg \textit{before} lunch”
- “Take your medicine \textit{before} lunch”
- “Finish lunch \textit{before} removing dishes”
Sometimes there is no solution

- “Introducing recursion requires that students know about stacks”

- “You must discuss abstract data types before introducing stacks”

- “Abstract data types rely on recursion”

The constraints introduce a cycle
Topological sort: example uses

- From a dictionary, produce a list of definitions such that no word occurs prior to its definition.

- Produce a complete schedule for carrying out a set of tasks, subject to ordering constraints. (This is done for scheduling maintenance tasks for industrial tasks, often with thousands of constraints.)

- Produce a version of a class with the features reordered so that no feature call appears before the feature’s declaration.
Overall structure (1)

Given:

- A type $G$
- A set of elements of type $G$
- A set of constraints between these elements

Required:

- An enumeration of the elements, in an order compatible with the constraints

```plaintext
class ORDERABLE [G] feature

    elements: LIST [G]

    constraints: LIST [TUPLE [G, G]]

    topsort: LIST [G] is
        ...
        ensure
            compatible (Result, constraints)

end
```
Some mathematical background...
Binary relation on a set

Any property that either holds or doesn’t hold between two elements of a set

On a set \( PERSON \) of persons, example relations are:

- **mother**: \( a \text{ mother } b \) holds if and only if \( a \) is the mother of \( b \)
- **father**
- **child**
- **sister**
- **sibling** (brother or sister)

Notation: \( a \, r \, b \) to express that \( r \) holds of \( a \) and \( b \).
Example: the \textit{before} relation

- “Remove dishes \textit{before} discussing politics”
- “Walk to Ütliberg \textit{before} lunch”
- “Take your medicine \textit{before} lunch”
- “Finish lunch \textit{before} removing dishes”

The set of interest:

\textit{Tasks} = \{\textit{Discuss, Lunch, Medicine, Remove, Ütliberg}\}

The constraining relation:

\textit{Dishes} \textbf{before} \textit{Politics}
\textit{Ütliberg} \textbf{before} \textit{Lunch}
\textit{Medicine} \textbf{before} \textit{Lunch}
\textit{Lunch} \textbf{before} \textit{Dishes}
Special relations on a set $X$

- **universal** $[X]$: holds between any two elements of $X$
- **id** $[X]$: holds between every element of $X$ and itself
- **empty** $[X]$: holds between no elements of $X$
We consider a relation $r$ on a set $P$ as a set of pairs in $P \times P$, containing all the pairs $[x, y]$ such that $x \ r \ y$.

Then $x \ r \ y$ simply means that $[x, y] \in r$.

Example:
Example: the *before* relation

- “Remove dishes *before* discussing politics”
- “Walk to Ütliberg *before* lunch”
- “Take your medicine *before* lunch”
- “Finish lunch *before* removing dishes”

The set of interest:

\[
Tasks = \{\text{Discuss, Lunch, Medicine, Remove, Ütliberg}\}
\]

The constraining relation:

\[
\text{constraints} = \\
\{[\text{Dishes, Politics}], [\text{Ütliberg, Lunch}],
\quad [\text{Medicine, Lunch}], [\text{Lunch, Dishes}]\}
\]
Using ordinary set operators

\[ \text{spouse} = \text{wife} \cup \text{husband} \]

\[ \text{sibling} = \text{sister} \cup \text{brother} \cup \text{id [Person]} \]

\[ \text{sister} \subseteq \text{sibling} \]

\[ \text{father} \subseteq \text{ancestor} \]

\[ \text{universal} [X] = X \times X \quad \text{(cartesian product)} \]

\[ \text{empty} [X] = \emptyset \]
Possible properties of a relation

(On a set \( X \). All definitions must hold for every \( a, b, c \ldots \in X \).)

- Total: \( (a \ r \ b) \lor (b \ r \ a) \)
- Reflexive: \( a \ r \ a \)
- Symmetric: \( a \ r \ b \ \Rightarrow \ b \ r \ a \)
- Antisymmetric: \( (a \ r \ b) \land (b \ r \ a) \ \Rightarrow \ a = b \)
- Transitive: \( (a \ r \ b) \land (b \ r \ c) \ \Rightarrow \ a \ r \ c \)
Examples (on a set of persons)

- **sibling**
  - Reflexive, symmetric, transitive

- **sister**
  - Symmetric, transitive

- **family_head**
  - Reflexive, antisymmetric
    
    \( (a \text{ family\_head} b \text{ means } a \text{ is the head of } b\text{'s family, with one head per family}) \)

- **mother**
  - (None)

- **Total:** \( (a \, r \, b) \lor (b \, r \, a) \)
- **Reflexive:** \( a \, r \, a \)
- **Symmetric:** \( a \, r \, b \Rightarrow b \, r \, a \)
- **Antisymmetric:**
  \( (a \, r \, b) \land (b \, r \, a) \Rightarrow a = b \)
- **Transitive:**
  \( (a \, r \, b) \land (b \, r \, c) \Rightarrow a \, r \, c \)
Total order relation

Relation is **total order** if:
- Total
- Reflexive
- Transitive
- Antisymmetric

- **Total:** \((a \, r \, b) \lor (b \, r \, c)\)
- **Reflexive:** \(a \, r \, a\)
- **Symmetric:** \(a \, r \, b \Rightarrow b \, r \, a\)
- **Antisymmetric:**
  \[(a \, r \, b) \land (b \, r \, a) \Rightarrow a = b\]
- **Transitive:**
  \[(a \, r \, b) \land (b \, r \, c) \Rightarrow a \, r \, c\]

Example: “less than equal” \(\leq\) on integers (or reals)

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>0 \leq 0,</td>
<td>0 \leq 1,</td>
<td>0 \leq 2,</td>
<td>0 \leq 3,</td>
<td>0 \leq 4, ...</td>
<td></td>
</tr>
<tr>
<td>1 \leq 1,</td>
<td>1 \leq 2,</td>
<td>1 \leq 3,</td>
<td>1 \leq 4, ...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 \leq 2,</td>
<td>2 \leq 3,</td>
<td>2 \leq 4, ...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Chair of Software Engineering*
Partial order relation

Relation is a **partial order** if:

- Reflexive
- Transitive
- Antisymmetric

- **Total:** \( a \mathrel{r} b \lor (b \mathrel{r} c) \)
- **Reflexive:** \( a \mathrel{r} a \)
- **Symmetric:** \( a \mathrel{r} b \Rightarrow b \mathrel{r} a \)
- **Antisymmetric:**
  \[
  (a \mathrel{r} b) \land (b \mathrel{r} a) \Rightarrow a = b
  \]
- **Transitive:**
  \[
  (a \mathrel{r} b) \land (b \mathrel{r} c) \Rightarrow a \mathrel{r} c
  \]

Example: relation between points in a plane:

\( p \sqsubseteq q \) if both:

- \( x_p \leq x_q \)
- \( y_p \leq y_q \)
Example partial order

Here the following hold:

\( a \preceq b \quad b \preceq d \quad c \preceq d \quad a \preceq d \)

No link between \( a \) and \( c \), \( b \) and \( c \):

\( a \not\preceq c \) nor \( a \not\preceq b \)

\( p \preceq q \) if both

\( x_p \leq x_q \)
\( y_p \leq y_q \)

\( [0, 1] \)
\( [1, 2] \)
\( [2, 3] \)
\( [3, 0] \)
\( [4, 2] \)

\( a \)
\( b \)
\( c \)
\( d \)
Possible topological sorts

\[ a \preceq b \quad b \preceq d \quad c \preceq d \quad a \preceq d \]
Topological sort understood

Here the relation $\subseteq$ is:

$\{[a, b], [a, d], [b, d], [c, d]\}$

One of the solutions is:

$\{[a, a], [a, b], [a, c], [a, d], [b, b], [b, c], [b, d], [c, c], [c, d], [d, d]\}$

We are looking for a total order relation $t$ such that $\subseteq \subseteq t$
Final statement of topological sort problem

From a given partial order, produce a compatible total order

where:

A partial order $p$ is compatible with a total order $t$ if and only if

$$p \subseteq t$$
The relation defined by a set of constraints, such as

\[
\text{constraints} = \\
\{ [\text{Dishes, Politics}], [\ddot{\text{U}}tliberg, \text{Lunch}], \\
[\text{Medicine, Lunch}], [\text{Lunch, Dishes}] \}
\]

is not by itself a partial order (although it’s easy to infer a partial order from it).
“Powers” and closure of a relation $r$

- $r^0 = \text{id} \ [X]$ where $X$ is the underlying set
- $r^{i+1} = r^i \ ; \ r$ where $;\,$ is composition

Transitive closure
- $r^+ = r^1 \cup r^2 \cup \ldots$ always transitive

Reflexive transitive closure
- $r^* = r^0 \cup r^1 \cup r^2 \cup \ldots$ always reflexive and transitive
No cycle in a relation

\[ r^+ \cap id \left[ X \right] = \emptyset \]
The partial order of interest is \textit{constraints}^*
Back to software...
Overall structure (1)

Given:

- A type \( G \)
- A set of elements of type \( G \)

Required:

- An enumeration of the elements, in an order compatible with the constraints

```plaintext
class ORDERABLE [G] feature

constraints: LIST [TUPLE [G, G]]

topsort: LIST [G] is

... ensure
compatible (Result, constraints)

end
```
In general there are several possible solutions.

In practice topological sort uses an optimization criterion to choose between possible solutions.
Overall structure (2)

\textit{topsort: LIST }[G]\textit{ is}

\textbf{require}

\hspace{1em} no\textunderscore cycle (constraints)

\hspace{1em} ...

\textbf{ensure}

\hspace{1em} compatible (Result, constraints)
Cycles

- must be a partial order: no cycle in the transitive closure of \textit{constraints}
  - No circular chain of the form $e_0 \rightarrow e_1, \ldots e_n \rightarrow e_0$

- If there are cycles there exists no solution to the topological sort problem!
Overall structure (2)

topsort: LIST [G] is
  require
    no_cycle (constraints)
  ...
  ensure
    compatible (Result, constraints)

Assume there are no cycles in the constraints
  • Not realistic since input may have errors
Don’t assume anything; find cycles as byproduct of attempt to do topological sort

“Attempt to do the topological sort, accounting for possible cycles in the constraints”

if “Cycles found” then
  “Report presence of one or more cycles in constraints”
end
End of lecture 24