“Topological sort”

From a given partial order, produce a compatible total order
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 Üetliberg 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
  - Üetliberg, Medicine, Lunch, Remove, Discuss: OK
  - Medicine, Üetliberg, Lunch, Remove, Discuss: OK
  - Discuss, Medicine, Lunch, Remove, Üetliberg: not OK

Pictured as a graph

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

```
class ORDERABLE [G] feature
  elements: LIST [G]
  constraints: LIST [TUPLE [G, G]]
  tosort: LIST [G] is ...
  ensure compatible (Result, constraints)
end
```
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 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 before relation

The set of interest:
Tasks = {Discuss, Lunch, Medicine, Remove, Ütliberg}

The constraining relation:
Dishes before Politics
Ütliberg before Lunch
Medicine before Lunch
Lunch before 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

Relations: a more precise mathematical view

- 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

The set of interest:
Tasks = {Discuss, Lunch, Medicine, Remove, Ütliberg}

The constraining relation:
constraints = {{Dishes, Politics}, [Ütliberg, Lunch], [Medicine, Lunch], [Lunch, Dishes]}

Using ordinary set operators

- spouse = wife ∪ husband
- sibling = sister ∪ brother ∪ id [Person]
- sister ⊆ sibling
- father ⊆ ancestor

universal [X] = X × X (cartesian product)
empty [X] = ∅
Possible properties of a relation

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

- Total: $(a \leq b) \lor (b \leq a)$
- Reflexive: $a \leq a$
- Symmetric: $a \leq b \Rightarrow b \leq a$
- Antisymmetric: $(a \leq b) \land (b \leq a) \Rightarrow a = b$
- Transitive: $(a \leq b) \land (b \leq c) \Rightarrow a \leq c$

Examples (on a set of persons)

- sibling: Reflexive, symmetric, transitive
- sister: Symmetric, transitive
- family_head: Reflexive, antisymmetric
- mother: (None)

Total order relation

Relation is total order if:

- Total
- Reflexive
- Transitive
- Antisymmetric

Example: "less than equal" $\leq$ on integers (or reals)

Partial order relation

Relation is a partial order if:

- Total
- Reflexive
- Transitive
- Antisymmetric

Example: relation between points in a plane: $p \leq q$ if both

Possible topological sorts

Here the following hold:

- $a \leq b \leq c \leq d$
- No link between $a$ and $c$, $b$ and $c$
- e.g. neither $a \leq c$ nor $a \leq b$
- $a \leq d$
**Topological sort understood**

Here the relation \( \preceq \) is:
\[
\{(a, b), (a, d), (b, b), (b, d), (c, c), (c, d), (d, d)\}
\]

One of the solutions is:
\[
\{(a, a), (a, b), (a, c), (a, d), (b, b), (b, c), (b, d), (c, d), (d, d)\}
\]

We are looking for a total order \( t \) such that \( \preceq \subseteq t \)

**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

\[ p \subseteq t \]

**From constraints to partial orders**

The relation defined by a set of constraints, such as

\[
\text{constraints} = \{(\text{Dishes, Politics}), (\text{Ütliberg, Lunch}), (\text{Medicine, Lunch}), (\text{Lunch, Dishes})\}
\]

is not by itself a partial order (although it’s easy to infer a partial order from it).

**“Powers” and closure of a relation \( r \)**

- \( r^0 = \text{id} [X] \)
- \( r^{i+1} = r^i \circ r \)

where \( X \) is the underlying set

where \( \circ \) is composition

Transitive closure
- \( r^+ = r^0 \cup r^1 \cup r^2 \cup \ldots \) 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**

\[ r^1 \cap \text{id} [X] = \emptyset \]

**From constraints to partial orders**

The partial order of interest is \( \text{constraints}^* \)

\[
\text{constraints}^* = \{(\text{Dishes, Politics}), (\text{Ütliberg, Lunch}), (\text{Medicine, Lunch}), (\text{Lunch, Dishes})\}
\]

Ütliberg

Medicine

Lunch

Dishes

Politics
Introduction to Programming – Lecture 24

**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
    require
        no_cycle (constraints)
    ... 
    ensure
        compatible (Result, constraints)
end
```

**Non-uniqueness**

- In general there are several possible solutions
- In practice topological sort uses an optimization criterion to choose between possible solutions.

**Overall structure (2)**

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

**Cycles**

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

- If there are cycles there exists no solution to the topological sort problem!

**Overall structure (2)**

```plaintext
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
Overall structure (3)

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