Discussion #25

Set Topics & Applications
Topics

• Sequences
• Strings
• Power Sets
• Types
• Relations — definitions & representations
Sequences

• Ordered list of elements from a set $A$
  – The sequence is said to be *over* $A$.
  – Repetitions of elements are allowed.
  – A sequence of *length* $n$ is an *$n$-tuple*.

• Delimited by angle brackets
  – e.g. $<a,b,c>$, $<a,c,b,a,b,d>$
  – Empty sequence $<>$
  – Infinite sequence $<1,1,2,2,1,1,2,2,1,1,\ldots>
Formal Notation for Sequences

• Alphabet A

• Cross-product notation
  – $A \times A = A^2$
  – $A \times \ldots \times A = A^n$

• Set of all nonempty sequences over A
  – Of length $\leq n$: $A^1 \cup A^2 \cup \ldots \cup A^n$
  – Of any length: $A^1 \cup A^2 \cup \ldots = A^+$
  – Including empty: $A^0 \cup A^1 \cup \ldots = A^*$, where $A^0 = \langle \rangle$
Strings

• Strings
  – Sequences of characters are called strings.
  – The set of characters over which a string is formed is called an alphabet.
  – Normally we omit the commas and use quotes:
    <a,b,c,a> = 'abca'

• Concatenation (\(\cdot\),\(+\))
  – Examples:
    • \(<1,3,5>,<2,3,3>\) = \(<1,3,5>+<2,3,3>\) = \(<1,3,5,2,3,3>\)
    • 'ab' \(\cdot\)'ca' = 'ab'+'ca' = 'abca'
  – Concatenation is not commutative.
  – Concatenation is associative.
Subsequences & Substrings

• Subsequences: if $S$ is a sequence, $X$ is a subsequence if $S = y + X + z$, where $y$ and $z$ are (possibly empty) sequences.

• Substrings: string 'ab' is a substring of 'caabb' and of 'ab' as well.
Power Sets

• Set of all subsets of a set $A$.
  - $A = \{1,2\}$
  - $P(A) = 2^A = \{\{\}, \{1\}, \{2\}, \{1,2\}\}$

• We note that each element of $A$ is either present (1) or not present (0). If we treat the elements of $A$ as a sequence, we get a bit-string that represents the set.
  - If $A = \{a,b,c,d\}$, we can order the elements in a sequence $<a,b,c,d>$.
  - Then, we can let the bit-string say which elements are present: e.g. 0110 means $\{b,c\}$.
  - We can represent all the subsets of $A$, from $\emptyset = 0000$ to $U = 1111$.

• This bit-string notation also helps us know the number of subsets in the powerset (just count in binary)
  - $2^{\#A}$, $2^{|A|} = 2 \cdot 2 \cdot \ldots \cdot 2$ ($|A|$ times)
  - This motivates the notation $2^A$ for the power set.
Bit-String Operations on Power Sets

• With bit string representations
  – Set intersection: \( \cap = \) pairwise \( \land \)
  – Set union: \( \cup = \) pairwise \( \lor \)
  – Set complement: \( \sim = \) bit complement
  – Set minus: \( - = \) mask out using 1’s = complement 2\(^{nd} \) operand and do pairwise \( \land \)

• E.g. using \( \{a,b,c,d\} \)
  – \( 1011 \cap 1101 = 1001 \) i.e. \( \{a,c,d\} \cap \{a,b,d\} = \{a,d\} \)
  – \( 1011 \cup 1101 = 1111 \) i.e. \( \{a,c,d\} \cup \{a,b,d\} = \{a,b,c,d\} \)
  – \( \sim 1011 = 0100 \) i.e. \( \sim \{a,c,d\} = \{b\} \)
  – \( 1010 - 1100 = 0010 \) i.e. \( \{a,c\} - \{a,b\} = \{c\} \)
Types

• A type is a pair of sets: (set of values, set of operations).
• A type declaration is sometimes called a signature.
• Kinds of types
  – Built-in types
    • int
    • float
    • Boolean - \{T,F\}
    • char - the available characters
  – Derived types: can be restricted (derived) using set builder notation (or its equivalent in some other syntax):
    \{x \mid x \in \mathbb{N} \land x \geq 1 \land x \leq 10\}, where \(\mathbb{N}\) represents the natural numbers
    = \{x \in \mathbb{N} \mid x \geq 1 \land x \leq 10\}
    = \{x : \mathbb{N} \mid x \geq 1 \land x \leq 10\}
    = \{x \mid x \geq 1 \land x \leq 10\}, where \(\mathbb{N}\) is understood to be the UofD
    = \{x \mid x \in \{1, \ldots, 10\}\}
Types (continued…)

• When the operations are “closed”, the operations yield only a single type (or sort).
• Many-sorted algebras
  – (integers, \{+, −, =, ≠\}): 2 = 3 is False (False \∉\ integers: two sorts, namely, integer & Boolean)
  – (strings, \{ +, convert_to_integer\}): convert_to_integer('125') = 125 (125 \∉\ strings: two sorts, namely strings & integers)
  – Type casting: conversion from one type to another
• Note about the project
  – We could have assigned each attribute in a scheme to have a type (be associated with a subset of the domain).
  – Substitutions for variables for an attribute would have then been limited to the declared subset.
Binary Relations

• Sets of ordered 2-tuples (pairs) with values selected from domains (sets)
• Formally, a relation $R$ from set $A$ to set $B$ is a set of pairs $(x,y)$ such that $x \in A$ and $y \in B$. We may write $R: A \leftrightarrow B$ to express this.
  – $R \subseteq A \times B$
  – If $(x,y) \in R$, we say that $x$ is $R$-related to $y$; we may also write $xRy$.
  – Example:
    • $\lt : \mathbb{N} \leftrightarrow \mathbb{N}$ (where $\mathbb{N}$ is the set of natural numbers)
    • $(2,3) \in \lt$ or $2 < 3$
  – Predicates also define relations
    $\lt: \{0,1,2\} \leftrightarrow \{-1,0,1,2\}$
    = $\{(x,y) \mid x \in \{0,1,2\} \land y \in \{-1,0,1,2\} \land x < y\}$
    = $\{ (0,1), (0,2), (1,2) \}$
Representations for Binary Relations

**Tables**

<table>
<thead>
<tr>
<th></th>
<th>&lt;</th>
<th>-1</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Matrices**

\[
\begin{pmatrix}
0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0
\end{pmatrix}
\]

**Graphs**
- directed graph or digraph
- directed arcs
Domains and Ranges for Binary Relations

Let $R = \{(0,1), (0,2), (1,2)\}$, then

- **Domain of $R = \text{dom } R = \text{dom}(R) = \{x \mid \exists y((x,y) \in R)\}\)**
  - i.e. $\{0,1\}$
  - like $\pi_x R$

- **Range of $R = \text{ran } R = \text{ran}(R) = \{y \mid \exists x((x,y) \in R)\}\)**
  - i.e. $\{1,2\}$
  - like $\pi_y R$

Diagram:
- Domain Space: 0, 1, 2
- Range Space: -1, 0, 1, 2
- Domain
  - 0 → 1 → 2
  - 0 → -1
  - 2 → 1 → 2