CSE 311 Foundations of Computing I

Lecture 9

Set Theory and Functions
Spring 2013

Announcements

- Reading assignments
  - Sets and Functions
    - 2.1-2.3 6th and 7th Editions
  - Monday: Modular Arithmetic
    - 4.1-4.2 7th Edition
    - 3.4, 3.6 up to p. 227 6th Edition

Important: Applications of Inference Rules

- You can use equivalences to make substitutions of any subformula
- Inference rules only can be applied to whole formulas (not correct otherwise).

e.g. 1. $p \to q$ Given

~~2. $(p \vee r) \to q$ Intro $\vee$ from 1.~~

Does not follow! e.g p=**F**, q=**F**, r=**T**

Set Theory

- Formal treatment dates from late 19th century
- Direct ties between set theory and logic
- Important foundational language

Definition: A set is an unordered collection of objects

$x \in A$: “x is an element of A”
“x is a member of A”
“x is in A”

$x \notin A$: $\neg (x \in A)$

Definitions

- A and B are *equal* if they have the same elements

$$A = B \equiv \forall x\,(x \in A \leftrightarrow x \in B)$$

- A is a *subset* of B if every element of A is also in B

$$A \subseteq B \equiv \forall x\,(x \in A \rightarrow x \in B)$$

Empty Set and Power Set

- Empty set $\emptyset$ does not contain any elements
- Power set of a set A = set of all subsets of A

$$\mathcal{P}(A) = \{ B : B \subseteq A\}$$

Cartesian Product : $A \times B$

$$A \times B = \{ (a, b) \mid a \in A \wedge b \in B\}$$

Set operations

$A \cup B = \{ x \mid (x \in A) \vee (x \in B) \}$ union

$A \cap B = \{ x \mid (x \in A) \wedge (x \in B) \}$ intersection

$A - B = \{ x \mid (x \in A) \wedge (x \notin B) \}$ set difference

$A \oplus B = \{ x \mid (x \in A) \oplus (x \in B) \}$ symmetric difference

$\overline{A} = \{ x \mid x \notin A \}$
(with respect to universe U) complement

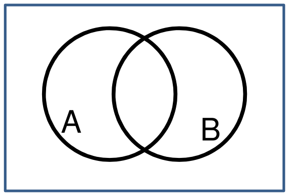
It’s Boolean algebra again

- Definition for ∪ based on ∨
- Definition for ∩ based on ∧
- Complement works like ¬

De Morgan’ s Laws

$$\overline{A \cup B} = \overline{A} \cap \overline{B}$$

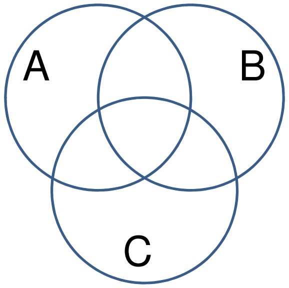
$$\overline{A \cap B} = \overline{A} \cup \overline{B}$$

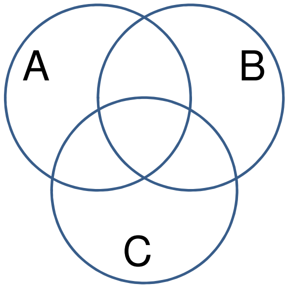


Proof technique:
To show C = D show
$x \in C \rightarrow x \in D$ and
$x \in D \rightarrow x \in C$

Distributive Laws

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$
$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$





Characteristic vectors: Representing sets using bits

- Suppose universe U is {1,2,...,n}
- Can represent set $B \subseteq U$ as a vector of bits:

$$b_1 b_2 \ldots b_n \text{ where } b_i=1 \equiv (i \in B)$$

$$b_i=0 \equiv (i \notin B)$$

  - Called the *characteristic vector* of set B
- Given characteristic vectors for A and B
  - What is characteristic vector for $A \cup B$? $A \cap B$?

Boolean operations on bit-vectors: (a.k.a. bit-wise operations)

- $$\begin{array}{r} 01101101 \\ \vee\ \underline{00110111} \\ 01111111 \end{array}$$ Java: `z=x|y`
- $$\begin{array}{r} 00101010 \\ \wedge\ \underline{00001111} \\ 00001010 \end{array}$$ Java: `z=x&y`
- $$\begin{array}{r} 01101101 \\ \oplus\ \underline{00110111} \\ 01011010 \end{array}$$ Java: `z=x^y`

A simple identity

- If x and y are bits: $(x \oplus y) \oplus y = ?$
- What if x and y are bit-vectors?

Private Key Cryptography

- Alice wants to be able to communicate message secretly to Bob so that eavesdropper Eve who hears their conversation, cannot tell what Alice’s message is.
- Alice and Bob can get together and privately share a secret key K ahead of time.

One-time pad

- Alice and Bob privately share random n-bit vector K
  - Eve does not know K
- Later, Alice has n-bit message m to send to Bob
  - Alice computes $C = m \oplus K$
  - Alice sends C to Bob
  - Bob computes $m = C \oplus K$ which is $(m \oplus K) \oplus K$
- Eve cannot figure out m from C unless she can guess K

Unix/Linux file permissions

- **`ls -l`**

```
drwxr-xr-x ... Documents/
-rw-r--r-- ... file1
```

- Permissions maintained as bit vectors
  - Letter means bit is 1 **–** means bit is 0.

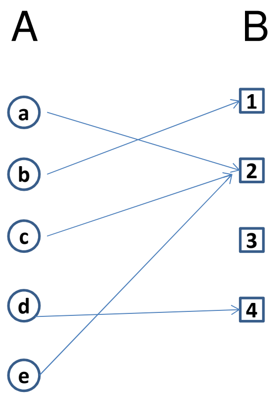
Russell's Paradox

$$\boxed{S = \{ x \mid x \notin x \}}$$

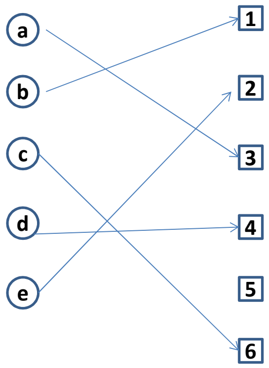
Functions review

- A *function* from *A* to *B*
  - an assignment of exactly one element of *B* to each element of *A.*
  - We write *f: A→B.*
  - "Image of *a"* = *f(a)*
- *Domain* of *f* : A
- *Range* of *f* = set of all images of elements of A

Image, Preimage



Is this a function? one-to-one? onto?


a
b
c
d
e
1
2
3
4
5
6