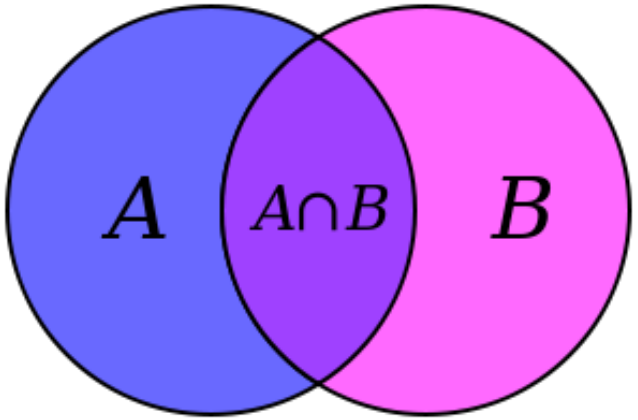
CSE 311



Foundations of Computing I

Fall 2014

It's Boolean algebra again

- Definition for $\cup$ based on $\vee$

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

- Definition for $\cap$ based on $\wedge$

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

- Complement works like $\neg$

$$\overline{A} = \{ x : x \notin A \}$$

(with respect to universe U)

De Morgan's Laws

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

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

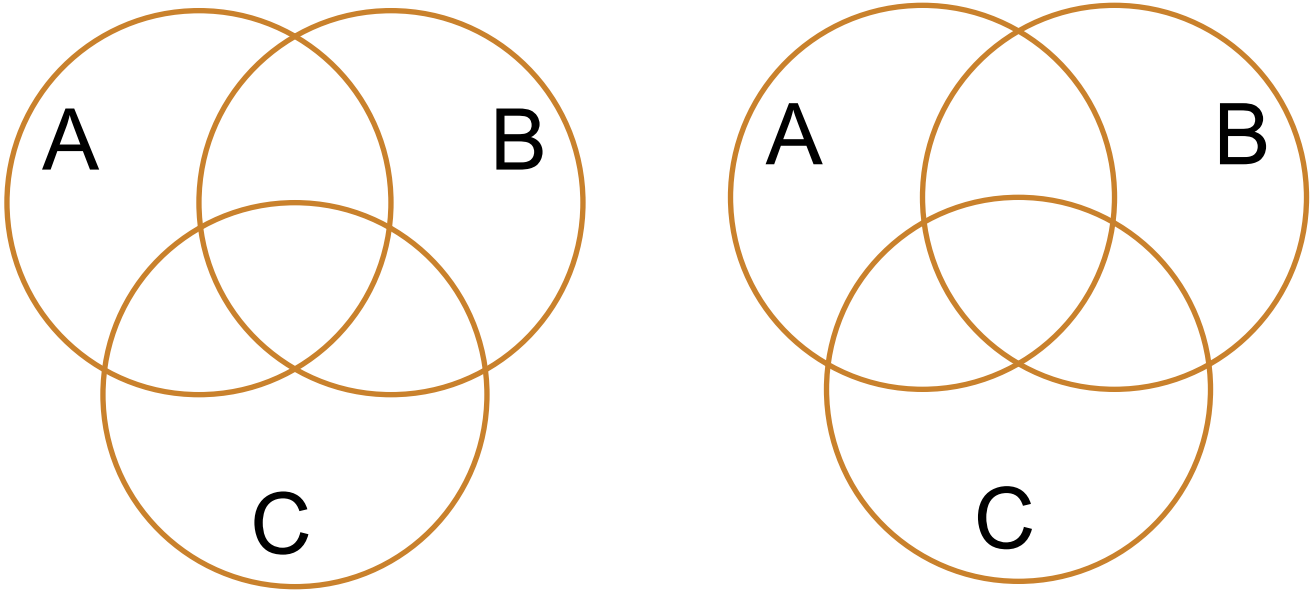
Let U be the universe.

$$\begin{aligned}
\overline{A \cup B} &= \{x : x \notin A \cup B\} \\
&= \{x : \neg(x \in A \cup B)\} \\
&= \{x : \neg((x \in A) \vee (x \in B))\} \\
&= \{x : (x \notin A) \wedge (x \notin B)\} \\
&= \{x : (x \in \overline{A}) \wedge (x \in \overline{B})\} \\
&= \{x : x \in \overline{A}\} \cap \{x : x \in \overline{B}\} \\
&= \overline{A} \cap \overline{B}
\end{aligned}$$

$$\begin{aligned}
x \in \overline{A \cap B} &\equiv x \notin A \cap B \\
&\equiv \neg(x \in A \cap B) \\
&\equiv \neg((x \in A) \wedge (x \in B)) \\
&\equiv (x \notin A) \vee (x \notin B) \\
&\equiv (x \in \overline{A}) \vee (x \in \overline{B}) \\
&\equiv x \in \overline{A} \cup \overline{B}
\end{aligned}$$

Distributive Laws

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



Representing Sets Using Bits

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

  $b_1 b_2 \ldots b_n$ where $b_i = 1$ when $i \in B$

  $b_i = 0$ when $i \notin B$

  – Called the *characteristic vector* of set B

- Given characteristic vectors for A and B
  – What is characteristic vector for $A \cup B$? $A \cap B$?

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.

Bitwise Operations

```
  01101101
∨ 00110111
----------
  01111111
```

Java: `z=x|y`

```
  00101010
∧ 00001111
----------
  00001010
```

Java: `z=x&y`

```
  01101101
⊕ 00110111
----------
  01011010
```

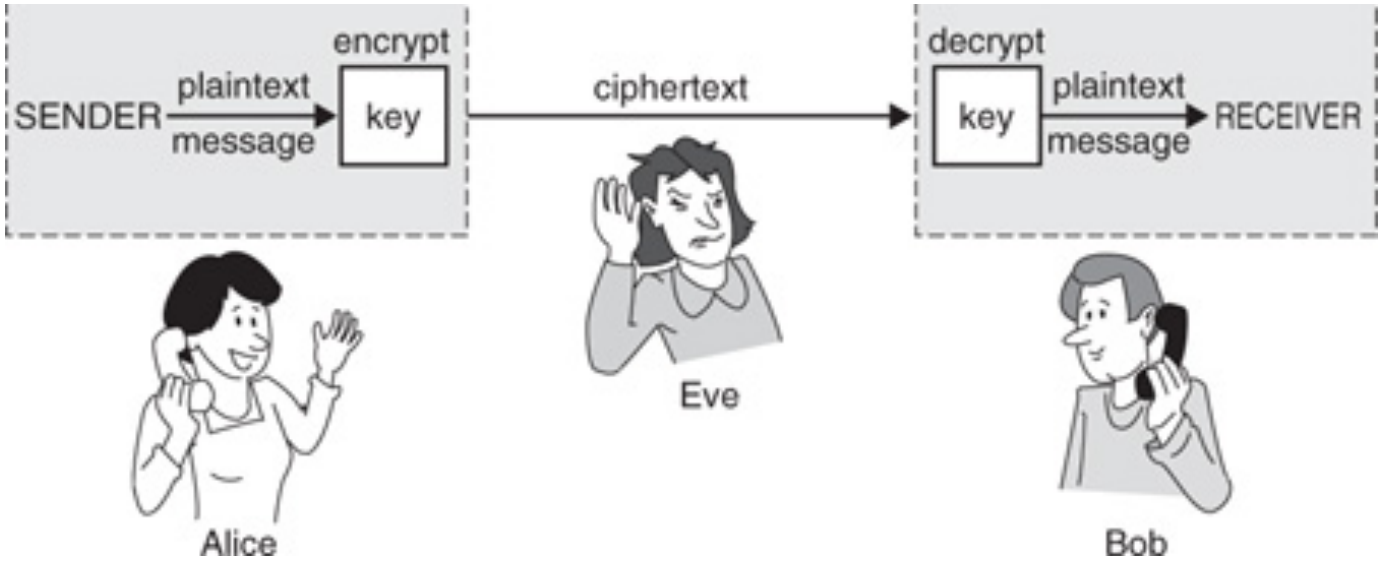
Java: `z=x^y`

A Useful Identity

- If x and y are bits: $(x \oplus y) \oplus y = ?$
  - $(x \oplus y) \oplus y = x \oplus (y \oplus y) = x \oplus 0 = x$

Private Key Cryptography

- **Alice** wants 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

CSE 311: Foundations of Computing

Fall 2014

Lecture 10: Functions, Modular arithmetic

Announcements

- Homework 3 due now
  - Homework 2 Solutions available
  - Homework 4 out later today

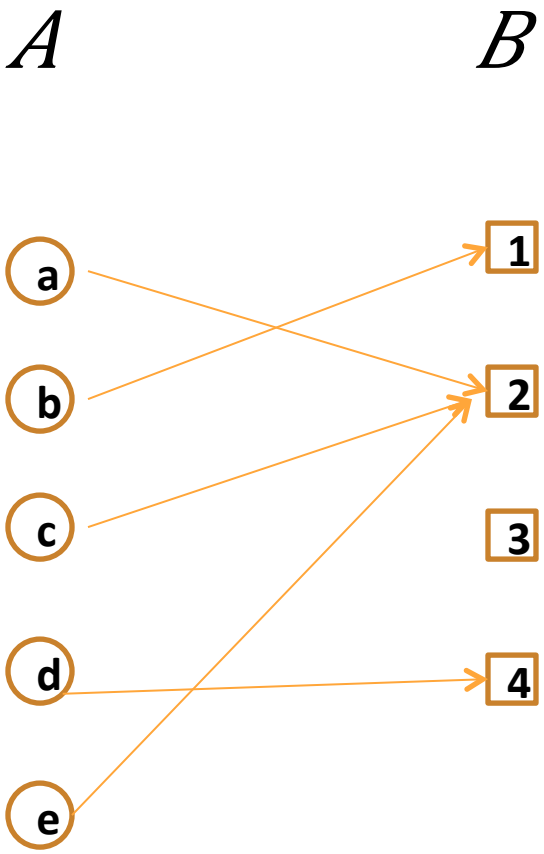
Functions

- A *function* from A to B.
  - Every element of A is assigned to exactly one element of B.
  - We write $f : A \rightarrow B$.
  - "Image of X" = $\{x : \exists y\ (y \in X \wedge x = f(y))\}$

- *Domain* of *f* is A
- Codomain of f is B

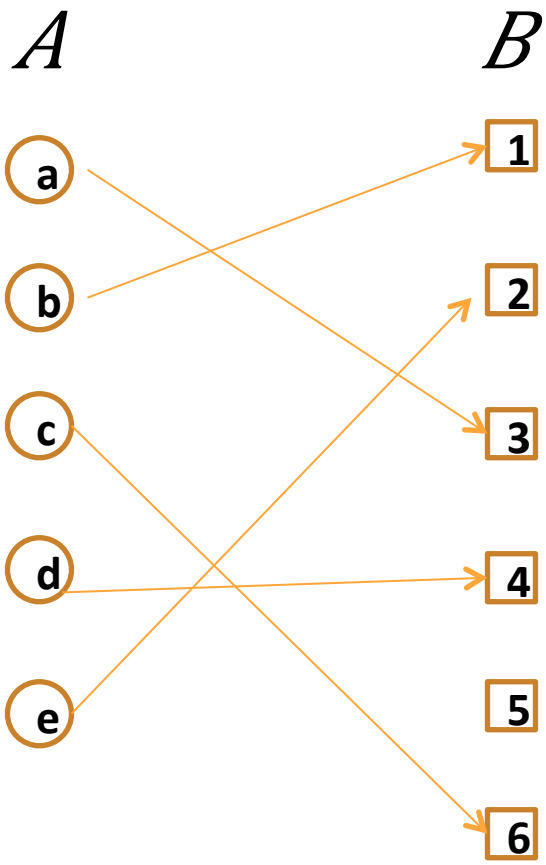
- *Image* of f = Image of domain = all the elements pointed to by something in the domain.

Image



Image({a}) = {2}
Image({a, e}) = {2}
Image({a, b}) = {1, 2}
Image(A) = {1, 2, 4}

Is this a function? One-to-One? Onto?



It is one-to-one, because nothing in B is pointed to by multiple elements of A.

It is not onto, because 5 is not pointed to by anything.

Functional Examples

Domain: Reals

	One-to-one	Onto
$x \mapsto x^2$	No (-1, 1)	No (no negatives)
$x \mapsto x^3 - x$	No (-1, 1)	Yes
$x \mapsto e^x$	Yes	No (no negatives)
$x \mapsto x^3$	Yes	Yes

Number Theory (and applications to computing)

- Branch of Mathematics with direct relevance to computing

- Many significant applications
  - Cryptography
  - Hashing
  - Security

- Important tool set

Modular Arithmetic

- Arithmetic over a finite domain

- In computing, almost all computations are over a finite domain

I’m ALIVE!

```
public class Test {
    final static int SEC_IN_YEAR = 364*24*60*60*100;
    public static void main(String args[]) {
        System.out.println(
            “I will be alive for at least ” +
            SEC_IN_YEAR * 101 + “ seconds.”
        );
    }
}
```

```
----jGRASP exec: java Test
I will be alive for at least -186619904 seconds.

----jGRASP: operation complete.
```

Divisibility

Integers a, b, with $a \neq 0$, we say that a *divides* b iff there is an integer k such that $b = ka$. The notation $a \mid b$ denotes "a divides b."

Division Theorem

Let a be an integer and d a positive integer. Then there are *unique* integers q and r, with $0 \leq r < d$, such that $a = dq + r$.

$q = a$ **div** d $\qquad\qquad$ $r = a$ **mod** d

Note: r ≥ 0 even if a < 0.
Not quite the same as **`a % d.`**

Division Theorem

Let *a* be an integer and *d* a positive integer. Then there are *unique* integers *q* and *r*, with $0 \le r < d$, such that $a = dq + r$.

$q = a$ **div** d $\qquad\qquad$ $r = a$ **mod** d

```
public class Test2 {
    public static void main(String args[]) {
        int a = -5;
        int d = 2;
        System.out.println(a % d);
    }
}
```

```
----jGRASP exec: java Test2
-1

----jGRASP: operation complete.
```

Note: r ≥ 0 even if a < 0.
Not quite the same as **`a % d.`**

Arithmetic, mod 7

$$a +_7 b = (a + b) \bmod 7$$

$$a \times_7 b = (a \times b) \bmod 7$$

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

x	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

Modular Arithmetic

Let a and b be integers, and m be a positive integer. We say a *is congruent to b modulo m* if m divides a – b. We use the notation a ≡ b (mod m) to indicate that a is congruent to b modulo m.

Modular Arithmetic: Examples

$A \equiv 0 \pmod 2$

This statement is the same as saying “A is even”; so, any A that is even (including negative even numbers) will work.

$1 \equiv 0 \pmod 4$

This statement is false. If we take it mod 1 instead, then the statement is true.

$A \equiv -1 \pmod{17}$

If $A = 17x - 1 = 17x + 16$, then it works.

Note that $(m - 1) \bmod m$

$= ((m \bmod m) + (-1 \bmod m)) \bmod m$

$= (0 + -1) \bmod m$

$= -1 \bmod m$

Modular Arithmetic: A Property

Let a and b be integers, and let m be a positive integer. Then $a \equiv b \pmod{m}$ if and only if $a \bmod m = b \bmod m$.

Proof: Suppose that $a \equiv b \pmod{m}$.

By definition: $a \equiv b \pmod{m}$ implies $m \mid (a - b)$ which by definition implies that $a - b = km$ for some integer k.

Therefore $a = b + km$. Taking both sides modulo m we get

$a \bmod m = (b + km) \bmod m = b \bmod m$.

Suppose that $a \bmod m = b \bmod m$.

By the division theorem, $a = mq + (a \bmod m)$ and

$b = ms + (b \bmod m)$ for some integers q,s.

$a - b = (mq + (a \bmod m)) - (mr + (b \bmod m))$

$= m(q - r) + (a \bmod m - b \bmod m)$

$= m(q - r)$ since $a \bmod m = b \bmod m$

Therefore $m \mid (a-b)$ and so $a \equiv b \pmod{m}$.