De Morgan’s Laws

\[ \overline{A \cup B} = \overline{A} \cap \overline{B} \]

Let \( U \) be the universe.

\[ \overline{A \cap B} = \overline{A} \cup \overline{B} \]

It’s Boolean algebra again

- Definition for \( \cup \) based on \( \lor \)
  \[ A \cup B = \{ x : (x \in A) \lor (x \in B) \} \]

- Definition for \( \cap \) based on \( \land \)
  \[ A \cap B = \{ x : (x \in A) \land (x \in B) \} \]

- Complement works like \( \neg \)
  \[ \overline{A} = \{ x : x \not\in A \} \]
  (with respect to universe \( U \))

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:
  \[ b_1 b_2 \ldots b_n \text{ where } b_i = 1 \text{ when } i \in B \]
  \[ b_i = 0 \text{ when } i \notin B \]
  - 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

<table>
<thead>
<tr>
<th></th>
<th>Java:</th>
</tr>
</thead>
<tbody>
<tr>
<td>01101101</td>
<td>( z = x</td>
</tr>
<tr>
<td>0110111</td>
<td>( 0111111 )</td>
</tr>
<tr>
<td>0010100</td>
<td>( 0000111 )</td>
</tr>
<tr>
<td>00110111</td>
<td>( z = x &amp; y )</td>
</tr>
<tr>
<td>01010101</td>
<td>( 00001010 )</td>
</tr>
<tr>
<td>00110111</td>
<td>( z = x \oplus y )</td>
</tr>
<tr>
<td>01101101</td>
<td>( 01011010 )</td>
</tr>
</tbody>
</table>

A Useful Identity

- If \( x \) and \( y \) are bits: \( (x \oplus y) \oplus y = ? \)
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 $X$” = $\{x : \exists y (y \in X \land x = f(y))\}$

- 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.

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

Functional Examples

<table>
<thead>
<tr>
<th>Domain: Reals</th>
<th>One-to-one</th>
<th>Onto</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x \mapsto x^2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x \mapsto x^3 - x$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x \mapsto e^x$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x \mapsto x^3$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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!

```java
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 + " seconds."
        );
    }
}
```
**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 $a = dq + r.$

$$q = \text{a div } d \quad r = \text{a mod } d$$

Note: $r \geq 0$ even if $a < 0$. Not quite the same as $a \% d$.

**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 $a \equiv b \pmod{m}$ to indicate that $a$ is congruent to $b$ modulo $m$.

**Arithmetic, mod 7**

$$a + 7 = (a + b) \mod 7$$

$$a \times 7 = (a \times b) \mod 7$$

<table>
<thead>
<tr>
<th>$+$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\times$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>
### Modular Arithmetic: Examples

<table>
<thead>
<tr>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A \equiv 0 \pmod{2} )</td>
<td>This statement is the same as saying “( A ) is even”; so, any ( A ) that is even (including negative even numbers) will work.</td>
</tr>
<tr>
<td>( 1 \equiv 0 \pmod{4} )</td>
<td>This statement is false. If we take it mod 1 instead, then the statement is true.</td>
</tr>
</tbody>
</table>
| \( A \equiv -1 \pmod{17} \) | If \( A = 17x - 1 = 17x + 16 \), then it works. Note that \( (m - 1) \mod m \)  
\[
= ((m \mod m) + (-1 \mod m)) \mod m  
= (0 + -1) \mod m  
= -1 \mod m
\] |

### 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 \( a \mod m = b \mod m \).