FINE 434: FinTech

Lecture 9

Professor Fahad Saleh

McGill University - Desautels

Equivalence Classes

We fix $N \in \mathbb{N}$ and define N equivalence classes, each populated by a subset of the integers. Equivalence class S_i is given by $S_i = \{n \in \mathbb{Z} : n \,\%\, N = i\}$ with $i \in \{0, 1, ..., N-1\}$ and $\%$ being the modulus operator.

Equivalence Classes

We fix $N \in \mathbb{N}$ and define N equivalence classes, each populated by a subset of the integers. Equivalence class S_i is given by $S_i = \{n \in \mathbb{Z} : n \,\%\, N = i\}$ with $i \in \{0, 1, ..., N-1\}$ and $\%$ being the modulus operator.

Suppose $N = 16$. Are 273 and 191 in the same equivalence class?

Equivalence Classes

We fix $N \in \mathbb{N}$ and define N equivalence classes, each populated by a subset of the integers. Equivalence class S_i is given by $S_i = \{n \in \mathbb{Z} : n \,\%\, N = i\}$ with $i \in \{0, 1, ..., N-1\}$ and % being the modulus operator.

Suppose $N = 16$. Are 273 and 191 in the same equivalence class?

```
1 273 % 16 == 191 % 16
```

```
False
```

Congruence Relation

Note that n is in the equivalence class $S_{i(n)}$ with $i(n) = n \,\%\, N$ and no other equivalence class. Thus, we can discuss the equivalence class for a given number without ambiguity.

Congruence Relation

Note that n is in the equivalence class $S_{i(n)}$ with $i(n) = n \,\%\, N$ and no other equivalence class. Thus, we can discuss the equivalence class for a given number without ambiguity.

We define a congruence relation, $\equiv$, over the set of integers. Concretely, we write $x \equiv z \ (mod\ N)$ if both x and z belong to the same equivalence class.

Note:
$x \equiv z \ (mod\ N)$
$\Leftrightarrow x \,\%\, N == z \,\%\, N$
$\Leftrightarrow (x - z) = k \cdot N$ for some integer k

Checking Congruence

Write a function that takes three integers, x, z and N, with $N > 0$ and returns True if and only if $x \equiv z \ (mod \ N)$

Checking Congruence

Write a function that takes three integers, x, z and N, with $N > 0$ and returns True if and only if $x \equiv z \ (mod\ N)$

```
1  import random
2
3  def isCongruent(x,z,N):
4      return(x%N == z%N)
5
6  for n in range(0,5):
7      N = random.randint(2,6)
8      x = random.randint(1,100)
9      z = random.randint(1,100)
10     if isCongruent(x,z,N):
11         print(str(x)+" is congruent to "+str(z)+" modulo "+str(N))
12     else:
13         print(str(x)+" is not congruent to "+str(z)+" modulo "+str(N))
```

```
76 is congruent to 2 modulo 2
48 is not congruent to 83 modulo 6
17 is congruent to 32 modulo 5
61 is not congruent to 38 modulo 4
16 is not congruent to 63 modulo 5
```

Equivalence Relation

- **Reflexivity**
  $x \equiv x \ (mod\ N)$
- **Symmetry**
  $x \equiv y \ (mod\ N) \Leftrightarrow y \equiv x \ (mod\ N)$
- **Transitivity**
  $x \equiv y \ (mod\ N)$ and $y \equiv z \ (mod\ N) \implies x \equiv z \ (mod\ N)$

Equivalence Relation

- **Reflexivity**
  $x \equiv x \ (mod\ N)$
- **Symmetry**
  $x \equiv y \ (mod\ N) \Leftrightarrow y \equiv x \ (mod\ N)$
- **Transitivity**
  $x \equiv y \ (mod\ N)$ and $y \equiv z \ (mod\ N) \implies x \equiv z \ (mod\ N)$

True or False?
$x \equiv y \ (mod\ N)$ and $z \equiv y \ (mod\ N) \implies z \equiv x \ (mod\ N)$

Basics

Assume $x_1 \equiv z_1 \ (mod\ N)$ and $x_2 \equiv z_2 \ (mod\ N)$. Then,

- **Translation**
  $x_1 + k \equiv z_1 + k \ (mod\ N)$ for any integer k
- **Scaling**
  $k \cdot x_1 \equiv k \cdot z_1 \ (mod\ N)$ for any integer k
- **Addition**
  $x_1 + x_2 \equiv z_1 + z_2 \ (mod\ N)$

What about subtraction?

Two Ways

$x_2 \equiv z_2 \ (mod\ N)$
$\implies -x_2 \equiv -z_2 \ (mod\ N)$ [Scaling]
$\implies x_1 - x_2 \equiv z_1 - z_2 \ (mod\ N)$ [Addition, $a - b = a + (-b)$]

$x_1 - x_2 \equiv z_1 - z_2 \ (mod\ N)$
$\Leftrightarrow (x_1 - x_2) - (z_1 - z_2) = k \cdot N$ for some integer k
$\Leftrightarrow (x_1 - z_1) - (x_2 - z_2) = k \cdot N$ for some integer k
$\Leftarrow (x_1 - z_1) = k_1 N$ and $(x_2 - z_2) = k_2 N$ for some integers k_1, k_2
$\Leftrightarrow x_1 \equiv z_1 \ (mod\ N)$ and $x_2 \equiv z_2 \ (mod\ N)$

Our Way

```
import random

def isCongruent(x,z,N):
    return(x%N == z%N)

subtractionWorks = True

for n in range(0,1000):
    N = random.randint(2,10)
    x1 = random.randint(-100,100)
    z1 = x1 + random.randint(-100,100)*N
    x2 = random.randint(-100,100)
    z2 = x2 + random.randint(-100,100)*N
    subtractionWorks = subtractionWorks and isCongruent(x1 - x2, z1 - z2, N)

if subtractionWorks:
    print("Subtraction is probably compatible with modular arithmetic")
else:
    print("Subtraction is not compatible with modular arithmetic")
```

```
Subtraction is probably compatible with modular arithmetic
```

Multiplication

Assume $x_1 \equiv z_1 \ (mod\ N)$ and $x_2 \equiv z_2 \ (mod\ N)$.

Then, $x_1 x_2 \equiv z_1 z_2 \ (mod\ N)$.

Multiplication

Assume $x_1 \equiv z_1 \ (mod\ N)$ and $x_2 \equiv z_2 \ (mod\ N)$.

Then, $x_1 x_2 \equiv z_1 z_2 \ (mod\ N)$. Why?

Multiplication

Assume $x_1 \equiv z_1 \ (mod\ N)$ and $x_2 \equiv z_2 \ (mod\ N)$.

Then, $x_1 x_2 \equiv z_1 z_2 \ (mod\ N)$. Why?

$x_1 x_2 \equiv z_1 z_2 \ (mod\ N)$
$\Leftrightarrow x_1 x_2 - z_1 z_2 = k \cdot N$ for some integer k
$\Leftrightarrow x_1 x_2 - z_1 x_2 + z_1 x_2 - z_1 z_2 = k \cdot N$ for some integer k
$\Leftrightarrow (x_1 - z_1) x_2 + z_1 (x_2 - z_2) = k \cdot N$ for some integer k
$\Leftarrow (x_1 - z_1) = k_1 N$ and $(x_2 - z_2) = k_2 N$ for some integers k_1, k_2
$\Leftrightarrow x_1 \equiv z_1 \ (mod\ N)$ and $x_2 \equiv z_2 \ (mod\ N)$

Corollaries

Assume $x \equiv z \ (mod \ N)$

Multiplication implies:
$x^2 \equiv z^2 \ (mod \ N)$

Corollaries

Assume $x \equiv z \ (mod \ N)$

Multiplication implies:
$x^2 \equiv z^2 \ (mod \ N)$

Induction implies:
$x^k \equiv z^k \ (mod \ N)$ for all $k \in \mathbb{Z}_+$

Corollaries

Assume $x \equiv z \ (mod \ N)$

Multiplication implies:
$x^2 \equiv z^2 \ (mod \ N)$

Induction implies:
$x^k \equiv z^k \ (mod \ N)$ for all $k \in \mathbb{Z}_+$

Scaling implies:
$a_k x^k \equiv a_k z^k \ (mod \ N)$ for all $k \in \mathbb{Z}_+, a_k \in \mathbb{Z}$

Corollaries

Assume $x \equiv z \ (mod \ N)$

Multiplication implies:
$x^2 \equiv z^2 \ (mod \ N)$

Induction implies:
$x^k \equiv z^k \ (mod \ N)$ for all $k \in \mathbb{Z}_+$

Scaling implies:
$a_k x^k \equiv a_k z^k \ (mod \ N)$ for all $k \in \mathbb{Z}_+, a_k \in \mathbb{Z}$

Addition implies:
$a_k x^k + a_l x^l \equiv a_k z^k + a_l z^l \ (mod \ N)$ for all $k, l \in \mathbb{Z}_+; a_k, a_l \in \mathbb{Z}$

Corollaries

Assume $x \equiv z \ (mod \ N)$

Multiplication implies:
$x^2 \equiv z^2 \ (mod \ N)$

Induction implies:
$x^k \equiv z^k \ (mod \ N)$ for all $k \in \mathbb{Z}_+$

Scaling implies:
$a_k x^k \equiv a_k z^k \ (mod \ N)$ for all $k \in \mathbb{Z}_+, a_k \in \mathbb{Z}$

Addition implies:
$a_k x^k + a_l x^l \equiv a_k z^k + a_l z^l \ (mod \ N)$ for all $k, l \in \mathbb{Z}_+; a_k, a_l \in \mathbb{Z}$

Induction implies:
$f(x) \equiv f(z) \ (mod \ N)$ for any integer polynomial function, f

One More

Assume $x \equiv z \ (mod \ N)$. Let k be any integer.

Does $k^x \equiv k^z \ (mod \ N)$ hold?

One More

Assume $x \equiv z \ (mod\ N)$. Let k be any integer.

Does $k^x \equiv k^z \ (mod\ N)$ hold?

```
import random

def isCongruent(x,z,N):
    return(x%N == z%N)

kraisedtoxworks = True

for n in range(0,1000):
    N = random.randint(2,10)
    x = random.randint(0,100)
    z = x + random.randint(0,100)*N
    k = random.randint(0,100)
    kraisedtoxworks = kraisedtoxworks and isCongruent(k**x, k**z, N)

if kraisedtoxworks:
    print("This is probably compatible with modular arithmetic")
else:
    print("This is not compatible with modular arithmetic")
```

```
This is not compatible with modular arithmetic
```

One More

Assume $x \equiv z \ (mod\ N)$. Let k be any integer.

Does $k^x \equiv k^z \ (mod\ N)$ hold?

```
import random

def isCongruent(x,z,N):
    return(x%N == z%N)

kraisedtoxworks = True

for n in range(0,1000):
    N = random.randint(2,10)
    x = random.randint(0,100)
    z = x + random.randint(0,100)*N
    k = random.randint(0,100)
    kraisedtoxworks = kraisedtoxworks and isCongruent(k**x, k**z, N)

if kraisedtoxworks:
    print("This is probably compatible with modular arithmetic")
else:
    print("This is not compatible with modular arithmetic")
```

```
This is not compatible with modular arithmetic
```

Alternatively, consider $N = 10$, $k = 2$, $x = 1$, $z = 11$

Existence of an Inverse Element

For any $e \in \mathbb{Z}$, there exists a $d \in \mathbb{Z}$ such that $e \cdot d \equiv 1 \ (mod\ N)$ if and only if $GCD(e, N) = 1$

Formally, let $S_{x,z} = \{n \in \mathbb{N} : x \ \% \ n == 0 \text{ and } z \ \% \ n == 0\}$. $S_{x,z}$ is non-empty, closed and bounded, so we may define $GCD(x, z) = \max S_{x,z}$.

$GCD(x, z)$ gives the greatest common divisor of x and z. $GCD(x, z) = 1$ means that x and z share no prime factors and therefore are deemed "relatively prime."

Euler's Theorem

Let $T_N = \{n \in \mathbb{N}_+ : GCD(n, N) = 1\}$. Define $\phi(N) = |T_N|$. Then, $\phi(N)$ gives the number of positive integers smaller than N that are relatively prime to N.

Euler's Theorem

Let $T_N = \{n \in \mathbb{N}_+ : GCD(n, N) = 1\}$. Define $\phi(N) = |T_N|$. Then, $\phi(N)$ gives the number of positive integers smaller than N that are relatively prime to N.

If $N = pq$ with p and q being prime then
$\phi(N) = N - p - q + 1 = (p-1)(q-1)$

Euler's Theorem

Let $T_N = \{n \in \mathbb{N}_+ : GCD(n, N) = 1\}$. Define $\phi(N) = |T_N|$. Then, $\phi(N)$ gives the number of positive integers smaller than N that are relatively prime to N.

If $N = pq$ with p and q being prime then
$\phi(N) = N - p - q + 1 = (p - 1)(q - 1)$

Theorem Statement: $x^{\phi(N)} \equiv 1 \ (mod\ N)$ for any $x \in T_N$

Chinese Remainder Theorem (sort of)

Assume p and q are both prime and $N = pq$

If $x \equiv z \ (mod\ p)$ and $x \equiv z \ (mod\ q)$ then $x \equiv z \ (mod\ N)$.

Chinese Remainder Theorem (sort of)

Assume p and q are both prime and $N = pq$

If $x \equiv z \ (mod\ p)$ and $x \equiv z \ (mod\ q)$ then $x \equiv z \ (mod\ N)$.

Actually, the Chinese Remainder Theorem is more general. The aforementioned statement follows from the full theorem. The result implies that establishing congruence equivalence among pairwise coprime factors suffices when establishing congruence equivalence with respect to the product of the pairwise coprime factors.