Groups and Rings - SF2729

Homework 2 (Rings)

Jim Holmström - 890503-7571

April 16, 2012

Exercise 1. Let $\sigma_m : \mathbb{Z} \to \mathbb{Z}_m$ be the natural homomorphism given by $\sigma_m(a) = a \ (mod\ m)$.

a. Show that $\overline{\sigma_m} : \mathbb{Z}[x] \to \mathbb{Z}_m[x]$ given by

$$\overline{\sigma_m}(a_0 + a_1x + \ldots + a_nx^n) = \sigma_m(a_0) + \sigma_m(a_1)x + \ldots + \sigma_m(a_n)x^n \quad (1)$$

is an homomorphism of $\mathbb{Z}[x]$ onto $\mathbb{Z}_m[x]$.

b. Show that $degree(f(x) \in \mathbb{Z}[x]) = degree(\overline{\sigma_m}(f(x))) = n \bigwedge \overline{\sigma_m(f(x))}$ has no nontrivial factors in $\mathbb{Z}_m[x] \Rightarrow f(x)$ is irreducible in $\mathbb{Q}[x]$.

c. Show that $x^3 + 17x + 36$ is irreducible in $\mathbb{Q}[x]$

Solution. a. $\overline{\sigma_m}(f(x)+g(x)) = \overline{\sigma_m} \sum (f_i + g_i)x^i = \sum \overline{\sigma_m}(f_i + g_i)x^i = \sum (\overline{\sigma_m}(f_i) + \overline{\sigma_m}(g_i))x^i = \overline{\sigma_m}(f(x)) + \overline{\sigma_m}(g(x))$ and $\overline{\sigma_m}(f(x)g(x)) = \overline{\sigma_m} \left(\sum \left(\sum f_i g_{n-i}\right) x^n\right) = \sum \overline{\sigma_m} \left(\sum f_i g_{n-i}\right) x^n = \sum \left(\sum \overline{\sigma_m}(f_i g_{n-i})\right) x^n = \sum \left(\sum \overline{\sigma_m}(f_i)\overline{\sigma_m}(g_{n-i})\right) x^n = \overline{\sigma_m}(f(x))\overline{\sigma_m}(g(x))$ Which shows that $\overline{\sigma_m}$ is an homomorphism.
$a(x) \in \mathbb{Z}_m[x]$ and $b(x) \in \mathbb{Z}[x]$ having the same coeffs but seen as in $\mathbb{Z}$instead of $\mathbb{Z}_m$ with this we see that $\overline{\sigma_m}(a(x)) = b(x)$, so it is onto. $\square$

b. $f = gh$ for $g, h \in \mathbb{Z}[x]$ where $degree(f) > degree(g) \wedge degree(f) > degree(h)$
Applying $\overline{\sigma_m}$ on f: $\overline{\sigma_m}(f) = \overline{\sigma_m}(g)\overline{\sigma_m}(h)$ is a factorization of $\overline{\sigma_m}$ into polynoms with a degree less then n of $\overline{\sigma_m}(f)$ which is a contradiction
$\Rightarrow f(x)$ is irreducible in $\mathbb{Z}[x]$
$\Rightarrow$ (by Theorem 23.11) $f(x)$ is irreducible in $\mathbb{Q}[x]$ $\square$

c. Magically choosing $m = 5$
$\overline{\sigma_5}(x^3 + 17x + 36) = x^3 + 2x + 1$
By hand it's simple to show that:

$$(x^3 + 2x + 1)(\{-2, -1, 0, 1, 2\}) \neq 0 \quad (2)$$

and by Theorem 23.10 irreducible over $\mathbb{Z}_5$ and by the findings in (b) we also have that $x^3 + 17x + 36$ is irreducible over $\mathbb{Q}$ $\square$

Exercise 2. Let $f(X) = X^4 - X^2 + 1$. Prove that $f(X)$ is irreducible in $\mathbb{Z}[X]$ and show that $f(X)$ is reducible in $\mathbb{Z}_m[X]$ for $m = \{2, 3, 5\}$ by determining the factorization into a product of irreducible polynomials.

Solution. Starting with the smaller rings:
m=2:
$(x^2 + x + 1)(x + 1)^2 = x^4 - x^2 + 1$ m=3:
$(x^2 + 1)^2 = x^4 - x^2 + 1$ m=5:
$(x^2 + 3x + 1)(x^2 + 2x + 4) = x^4 - x^2 + 1$ Which are all found by a computer program (see last in document) and hand verified to so that the calculations is correct and the terms indeed irreducible.

The computer-program in use:

```
import operator
import copy
import itertools as itt
import string
import math

#=======Ring definition==================
class Zn:
    def __init__(self,n,i):
        """
        Initz Z_n with the element i
        """
        self.n=n
        self.i=i%n #fugly but works with negative numbers which is nice (but platform depe

    def __str__(self):
        """
        You are on your own on tracking n, mostly one has the same n
        """
        return str(self.i)

    def __eq__(self,other):
        """
        NOOOOT!! Must be of the same Zn to be the same
        """
        if(isinstance(other,int)):
            return self.i==other
        return self.i==other.i
```

```
    def __ne__(self,other):
        return not operator.__eq__(self,other)

    def __add__(self,other):
        assert self.n==other.n
        return Zn(self.n,(self.i+other.i)%self.n)
    def __mul__(self,other):
        assert self.n==other.n #not defined else
        return Zn(self.n,(self.i*other.i)%self.n)
    def __pow__(self,m):
        """
        return g**n
        """
        return Zn(self.n,(self.i**m)%self.n)

    def __hash__(self):
        return self.i

class Polynom:
    def __init__(self,c):
        """
        Starts with the constant c_0
        """
        self.c=c

    def __str__(self):
        output=""
        for i,c_i in enumerate(reversed(self.c)):
            if((len(self)-i-1)>1): #fugly with double reverse TODO fix
                output+=str(c_i)
                output+="X^"+str(len(self)-i-1)+"+"
            elif((len(self)-i-1)==1):
                output+=str(c_i)+"X+"
            else:
                output+=str(c_i)
        return output

    def __eq__(self,other):
        N=len(self)
        M=len(other)
        #if all elements is equal and the part hanging outside is all zero then the polyno
        pseudoeq=all(map(lambda (a,b):a==b,zip(self.c,other.c)))
        if(N==M):#fugly code #TODO fixit
            return pseudoeq
```

```
        elif(M<N):
            return pseudoeq and reduce(operator.add,self.c[M:N])==0
        else:#(N<M)
            return pseudoeq and reduce(operator.add,other.c[N:M])==0

    def __neq__(self,other):
        return not operator.__eq__(self,other)

    def __add__(self,other):
        c_res=map(lambda (i,j):i+j,itt.izip_longest(self.c,other.c,fillvalue=0))
        return Polynom(c_res)

    def __len__(self):
        return len(self.c)

    def __mul__(self,other):
        #cauchy
        c_n=[0]*(len(self)+len(other)-1)

        for k in range(len(c_n)): #a little bit fugly with a forloop but gets nasty withou
            filtered=filter(lambda (i,j):i+j==k,itt.product(range(len(self)),range(len(oth
            c_n[k] = reduce(operator.add,map(lambda (i,j): self.c[i]*other.c[j],filtered))

        return Polynom(c_n)

degree=3
for m in [2,3,5]:
    print "m=",m,",degree",degree,":"
    Zm=map(lambda i:Zn(m,i),range(m))

    PZm=map(lambda c:Polynom(c),itt.product(Zm,repeat=(degree+1)))

    F=Polynom(map(lambda i:Zn(m,i),[1,0,-1,0,1])) #fugly since -1%m can be machinedependan

    factors=filter(lambda (f,g):f*g==F,itt.product(PZm,repeat=2))

    for f,g in factors:
        print str(f)+"*"+str(g)+"="+str(f*g)

#also tested the choosen factors from above from factors (to be irreducible)
```