CRC检验算法推导

从文档《CRC 算法原理及 C 语言实现》中我们得到

$$\frac{R_n \cdot 2 + B_{n-1} \cdot 2^8}{G(X)} = Q_{n-1} + \frac{R_{n-1}}{G(X)} \quad (3\text{-}5)$$

这里我把2^{16}换成了2^8，原理不变，简化运算量，所求出的公式也是针对8位CRC而言

这个式子说明，R_{n-1}可以由R_n和$B_{n-1} \cdot 2^{n-1}$来求得，具体的算法就是

R_n乘以2加上$B_{n-1} \cdot 2^8$，然后除以G(X)，其余数即为R_{n-1}

即，要求得下一位数据的CRC余数，可以由上一位数据的CRC余数乘以2然后再加上这一位数据乘以2的8次方，再除以多项式G(X)

$$R_{n-1} = R_n \times 2 + B_{n-1} \times 2^8 - Q_{n-1} \times G(X) \quad (4\text{-}1)$$

Q_{n-1}表示这位数据除以多项式G(X)的商，由于G(X)的最高位为1，Q_{n-1}的取值由$\left(R_n + B_{n-1} \times 2^7\right) \times 2$的最高位来决定

当它的最高位为1时，Q_{n-1}为1，当它的最高位为0时，Q_{n-1}为0

即$Q_{n-1} = R_{n7} + B_{n-1}$ (4-2)

其中R_{n7}表示R_n的第七位

把式4-2代入式4-1，得到

$$R_{n-1} = R_n \times 2 + B_{n-1} \times 2^8 - \left(R_{n7} + B_{n-1}\right) \times G(X)$$

这里将R_n拆开成为$R_{n7} \times 2^7$与$R_{n(0-6)}$两部分，将G(X)拆开成为$G_8 \times 2^8$与G_{0-7}两部分，代入上式中，得到

$$R_{n-1} = R_{n7} \times 2^8 + R_{n(0-6)} \times 2 + B_{n-1} \times 2^8 - \left(R_{n7} + B_{n-1}\right) \times \left(G_8 \times 2^8 + G_{0-7}\right)$$

$$R_{n-1} = \left(R_{n7} + B_{n-1}\right) \times 2^8 + R_{n(0-6)} \times 2 - \left(R_{n7} + B_{n-1}\right) \times G_8 \times 2^8 - \left(R_{n7} + B_{n-1}\right) \times G_{0-7}$$

$$R_{n-1} = \left(R_{n7} + B_{n-1} - \left(R_{n7} + B_{n-1}\right) \times G_8\right) \times 2^8 + R_{n(0-6)} \times 2 - \left(R_{n7} + B_{n-1}\right) \times G_{0-7}$$

由于G_8恒为1，因此$\left(R_{n7} + B_{n-1} - \left(R_{n7} + B_{n-1}\right) \times G_8\right) \times 2^8$恒为0

$$R_{n-1} = R_{n(0-6)} \times 2 - \left(R_{n7} + B_{n-1}\right) \times G_{0-7}$$

由于CRC采用GF(2)的运算法则，即不进位的加减法，上式中的加减运算可以转化成为异或运算

$$R_{n-1} = R_{n(0-6)} \times 2 \oplus \left(R_{n7} \oplus B_{n-1}\right) \times G_{0-7} \quad (4\text{-}3)$$

R_{n-1}表示当前位的CRC值

$R_{n(0-6)}$表示上一位的CRC值的第0至第6位

R_{n7} 表示上一位的CRC值的第7位

B_{n-1}表示当前位的值

G_{0-7}表示CRC多项式的第0至第7位

用代码来实现

$$R_{n-1} = R_{n(0-6)} \times 2 \oplus \left(R_{n7} \oplus B_{n-1}\right) \times G_{0-7} \quad (4\text{-}3)$$

上式可以化简成为两种情况

当$\left(R_{n7} \oplus B_{n-1}\right)$为1时，$R_{n-1} = R_{n(0-6)} \times 2 \oplus G_{0-7}$ 情况A

当$\left(R_{n7} \oplus B_{n-1}\right)$为0时，$R_{n-1} = R_{n(0-6)} \times 2$ 情况B

```
uint8_t crc = 0; //定义上一位的CRC值，初始化为0
uint8_t poly = 0x31; //CRC多项式的0至7位，省略掉了最高位
uint8_t data = 0xda; //待求CRC值的数据
int cnt = 0; //数据的位数
int i;
for(i=0;i<cnt;i++){
    uint8_t Rn7 = crc&0x80;  // 得到上次CRC值的第7位
    uint8_t Bn_1 = data&0x80; //得到当前待求的数据
    if(Rn7 ^ Bn_1){
        // 情况A
        crc<<=1; // 上次CRC值的0至6位乘2，因为crc是8位的，第7位被消掉
        crc = crc ^ poly; // 上次CRC值乘2后与多项式的0至7位异或
    }else{
        // 情况B
        crc <<= 1; // 上次CRC值的0至6位乘2，因为crc是8位的，第7位被消掉
    }
    data <<=1; //数据向左移
}
```

化简上面循环中的代码得到

```
for(i=0;i<cnt;i++){
    uint8_t Rn7 = (crc^data)&0x80;
    crc<<=1; // 上次CRC值的0至6位乘2，因为crc是8位的，第7位被消掉
```

```
    if(Rn7){
        // 情况A
        crc = crc ^ poly; // 上次CRC值乘2后与多项式的0至7位异或
    }
}
```