有限集拓扑的最简基研究

崔贵林

March 12, 2010

Abstract

本文主要研究在有限集中通过最简基生成拓扑的过程和其算法实现。

keywords:有限集，拓扑，最简基，算法

1 拓扑的基

定义：如果X是一个集合，X的某拓扑的一个基(basis)是X的子集的一个族$\mathcal{B}$(其成员称为基元素(basis element))，满足条件：
(1) 对于每一个$x \in X$，至少存在一个包含x的基元素B.
(2) 若x属于两个基元素B_1和B_2的交，则存在包含x的一个基元素B_3，使得$B_3 \subset B_1 \cap B_2$.
由基生成拓扑的另一种方法由以下引理给出。
引理：设X是一个集合，$\mathcal{B}$是X的拓扑τ的一个基。则τ等于$\mathcal{B}$中元素所有并的族。从拓扑基和基元素的定义里我们发现，没有对空集有特别要求，即空集可以是基元素，也可以不是基元素，无妨。但在引理中，说由基生成的拓扑等于基中元素所有并的族。如果这个并可以从零个并开始，并且说零个集合的并是空集的话，那么基中可以没有空集，否则基中必须有空集。否则空集由哪些基元素并出来呢？显然某一拓扑的拓扑基不唯一。

2 最简基

裴惠生在《河南大学学报》1987年第四期“关于有限集的拓扑种类问题”的论文中提出了最简基的概念。定义：设$X \neq \emptyset$,W为X上某一拓扑的基，若$\emptyset \notin W$,且$\forall B \in W$不存在$W' \subset W - \{B\}$使

$$B = \bigcup_{B' \in W'} B'$$

，则称W为X的一个最简基。最简基里明确了不包含空集。从这个最简基的定义里我们能够得出这样一个结论：令$\mathcal{W}$表示集合X的一拓扑τ的所有拓扑基的全体。则有：
W是X的拓扑τ的最简基

$$\Longleftrightarrow W = \bigcap_{W' \in \mathcal{W}} W'$$

换句话说，最简基也就是最小的拓扑基。
有了最简基的概念以后，拓扑和拓扑基之间就建立起一一对应关系了。
裴惠生老师在论文里定义X_n为n点集。这里再定义所有n点集的全体为$\mathcal{X}_n$
下面详细研究如何从最简基生成一个拓扑。

3 最简基生成拓扑

从最简拓扑基生成拓扑的步骤：

1. 空集$\emptyset$是开集，但不是最简基元素。

2. $\forall x \in X_n$，判断单点集x是否是开集？若x是开集，则x是最简基元素

3. $\forall x \in X_n, y \in X_n$，分两种情况：$\alpha$，若$x$和$y$都是开集，则可以直接得出：1，$x, y$是开集2，$x, y$不是最简基元素；$\beta$，若$x$和$y$至少有一个不是开集，则判断两点集x,y是否是开集？若x,y是开集，则x,y是最简基元素。记这样的两点集x,y的全体为集族$\mathcal{X}_2$。

4. $\forall X \in \mathcal{X}_2, Y \in \mathcal{X}_2$，令$Z = X \cup Y$。分两种情况：$\alpha$，若X和Y都是开集，则可以直接得出：1，Z是开集。2，Z不是最简基元素；β，若X和Y不全是开集，则判断Z是否是开集？若Z是开集，则Z是最简基元素。这里任意两个集合X和Y的并集Z可能有两种情况：1，$Z \in \mathcal{X}_3$，这时若Z第一次出现，则判断之，否则参照之前盼到你，而不必再判断。2，$Z \in \mathcal{X}_4 4$，这时不会重复出现，可直接判断。

5. 仿照上述步骤重复下去，假设我们已得到了k元集的全体$\mathcal{X}_k$。对于$\forall X \in \mathcal{X}_k, Y \in \mathcal{X}_k$，令$Z = X \cup Y$，这时首先看看Z是否已经被确定为开集。因为在$k \geq 2$时就可以跨级生成了。即二元集和二元集不只生成三元集，还生成了四元集。事实上，这里k元集生成的集合的基数是k+1到2k，当然还有一个条件，就是不会超过全集的个数n。好，如果足够幸运，Z是第一次生成的，那么和前面一样，分两种情况判断，这里不再多说。并集Z的生成情况如前所述，集合基数的范围是$[k+1, min(2k, n)] \cap N$.

6. 当判断到元素个数为n的时候，也就是全集X_n，这时有$\mathcal{X}_n = X_n$。如果X_n早已被$\mathcal{X}_k$的某两个开元素并出来，那么X_n不是最简基元素，否则X_n是最简基元素。无论哪种情况，X_n是开集，结束。

经过这样的步骤以后，一个拓扑，伴随着其最简基的确定，也就确定下来了。
其实这个过程类似二项展开，首先是空集，其个数是C_n^0，然后是单点集，这样的集合有C_n^1个，由单点集并出来的二元集的全体和n元集的幂集中的二元集的全体是一样的，个数是C_n^2。后面的依次类推。

4 确定拓扑的最简基

对于一个确定的拓扑，如何找寻它的最简基呢？先看一般的拓扑基。
引理：设X是一个拓扑空间，$\mathcal{C}$是X的开集的一个族，它满足对于X的每一个开集U以及每一个$x \in U$，存在$\mathcal{C}$的一个元素C，使得$x \in C \subset U$，那么$\mathcal{C}$就是X上这个拓扑的一个基。这是如何由给定的拓扑得到基的一种方

法，但不具有可操作性，另外拓扑基也不唯一。下面研究如何由拓扑得到最简基。
其实这就是对于拓扑中的每一个开集，判断它是否是最简基元素的过程。最简基元素区别于一般基元素或开集的本质属性在于什么呢？这是问题的关键。我们知道，开集可以是基元素，基元素可能是最简基元素。最简基元素一定是基元素，基元素一定是开集。一个开集，在特定的集合里，它就可以成为基元素，如果这个基元素又具备了某种性质，它就可以是最简基元素。基$\mathcal{C}$，最简基$\mathcal{W}$，拓扑τ的关系是这样的：

$$\mathcal{W} \subset \mathcal{C} \subset \tau$$

如何判断一个开集是不是最简基元素?
引理：设(X,τ)是拓扑空间，W是开集，如果$\exists x \in W$,不$\exists B \in \tau$，使得$x \in B$，则W是最简基元素。
显然。
这个引理比较具有可操作性。可以使用递归判断。
其实也可以使用上面最简基生成拓扑的过程来判断得到最简基。

5 编程实现

对于n点集X_n，使用长度为2^n的数组存储其拓扑结构。对于$\forall X \subset X_n$，X在数组中的存储位置为它的序号。这里定义一个映射$f : 2^X \to N$
下面程序初步实现了有限集的拓扑的子集的遍历，计算出每个子集的序号。
下一步将设置三种状态：0 闭集1 最简基元素2 开集

```
class Topology
{
    public:
        Topology(int n)
        {
            this->n = n;
            int n2=1;
            for(int i=0;i<n;i++)
                n2=n2*2;
            a = new int[n2];
            for(int i=0;i<n2;i++)
                a[i]=0;
            b = new int[n];
```

```
        b[0]=0;
    }
    void init()
    {
        f(n);
    }
protected:
    void f(int m)
    {
        if(m>1)
        {
            f(m-1);
        }
        g(1,m,m);
    }
    void g(int start,int r,int m)
    {
        if(r>0)
        {
            for(int i=start;i<n-r+2;i++)
            {
                b[m-r+1]=i;
                g(i+1,r-1,m);
            }
        }
        else
        {
            for(int i=1;i<m+1;i++)
                cout<<b[i]<<" ";
            this->r = m;
            cout<<"order is "<<order()<<endl;
        }

    }
protected:
    int order()
    {
        int order=0;
        for(int i=0;i<r;i++)
            order = order + Cnm(n,i);
        for(int j=1;j<r+1;j++)
```

```
            {
                for(int k=0;k<b[j]-b[j-1]-1;k++)
                    order = order + Cnm(n-b[j-1]-1-k,r-
                        j);
            }
            return order;
        }
        int Cnm(int n,int m)
            {
            int cnm=1;
            for(int i=n;i>n-m;i--)
                cnm=cnm*i;
            for(int i=2;i<m+1;i++)
                cnm=cnm/i;
            return cnm;
        }
    private:
        int n;
        int r;
        int *a;
        int *b;
};
```

References

[1] 拓扑学,美James R.Munkres著，熊金城，吕杰，谭枫译,机械工业出版社,2006年

[2] 关于有限集的拓扑种类问题,裴惠生,河南大学学报,1987年第四期