第六章面向源代码的软件可信性度量模型

陈仪香

2024年11月11日

Outline

第六章面向源代码的软件可信性度量模型

- 度量理论提供了一种用数值来刻画实体属性直觉性质的框架。
- 本章将度量理论应用于软件可信性度量中，基于度量理论中Extensive结构建立面向源代码的软件可信性度量模型，并利用公理化方法对所建软件可信性度量模型合理性进行理论验证，从而得到暨经度量理论验证又经公理化方法验证的软件可信性度量模型。
- 通过一个简单案例表明该度量模型的可用性和有效性。

6.1 经验关系系统

经验关系系统

Definition (经验关系系统)

给定一个属性，经验关系系统是一个有序元组

$$ERS = (E, R_1, \cdots, R_m, o_1, \cdots, o_n)$$

其中，

1. E：度量其属性的实体非空集；
2. $R_i(1 \leq i \leq m)$：集合E上的k_i元经验关系，用来表示关于属性的直觉知识；
3. $o_j(1 \leq j \leq n)$：集合E上的二元运算，即$o_j : E \times E \to E$。

经验关系系统

比如，假定我们要度量程序，则建立关于程序的经验关系系统

$$ERS_P = (E, \gg, ;)$$

其中，

1. E 为所有程序组成的集合；
2. $\gg \subseteq E \times E$表示关于程序经验理解的二元关系：给定两个程序e_1和e_2，若程序e_2是程序e_1的子程序，则我们认为$e_1 \gg e_2$，
3. 二元运算；：$e_3 = e_1; e_2$表示程序e_1和e_2的顺序连接。

经验关系系统

- 同一个实体集的其它属性，比如，复杂性，又不同于长度所具有的经验关系和操作，这是因为对于不同属性我们有不同经验理解。
- 经验关系系统只是用来人们关于实体属性的经验理解，而没有数或值的概念，因此也没有度量概念。
- 下面引入数值关系系统，用来进行度量引入。

6.2 数值关系系统

数值关系系统

Definition (数值关系系统)

给定一个属性，数值关系系统是一个有序元组

$$NRS = (V, S_1, \cdots, S_m, \oplus_1, \cdots, \oplus_n)$$

其中，

1. V：度量属性的度量值集；
2. $S_i(1 \leq i \leq m)$：集合V上的k_i元经验关系；
3. $\oplus_j(1 \leq j \leq n)$：集合$V$上的二元运算，即$\oplus_j$：$V \times V \to V$。

数值关系系统中V的元素一般是通常意思上的非负实数，但有时也可以是由一些A、B、C、D之类的符号组成。

数值关系系统

比如，在上面提到的程序长度例子中，我们可以定义其程序长度数值关系系统

$$NRS_{PL} = (V_I, \geq, +)$$

其中，

1. V_I是所有正整数组成的集合；
2. 二元关系为整数的普通大小关系$\geq$；
3. 二元操作为整数间普通加法$+$。

数值关系系统

再比如，定义有关程序的缺陷数值关系系统$NRS_{PE} = (V_e, \sqsubseteq, \wedge)$，其中，

- V_e是程序分类集合$\{A, B\}$，A表示程序无缺陷，而B表示程序有缺陷；
- V_e上的二元关系$\sqsubseteq$ 定义为$A \sqsubseteq B$，表示有缺陷程序的缺陷程度是大于等于无缺陷程序的缺陷程度；
- $\wedge$是集合V_e上的二元运算：

$$A \wedge A = A, B \wedge B = B, A \wedge B = B$$

表示有缺陷程序与任何程序合并都是有缺陷的。

数值关系系统的建立是为了将经验关系系统反映到值上，因此需要建立从经验关系系统到数值关系系统的联系，即，测量映射。

6.3 测量映射

测量映射

Definition (测量映射(Measure))

设$ERS = (E, R_1, \cdots, R_m, o_1, \cdots, o_n)$是一个经验关系系统，$NRS = (V, S_1, \cdots, S_m, \oplus_1, \cdots, \oplus_n)$是一个数值关系系统。任何一个从$E$到$V$的函数$\mu : E \to V$ 都称为测量映射。

- 对于程序经验关系系统ERS_P和程度数值关系系统NRS_{PL}，我们定义一个测量函数$\mu_L : E \to V_I$：给定一个程序e，$\mu_L(e)$ =程序e的代码行数。
- 同样，对于程序缺陷数值关系系统NRS_{PE}, 我们也可以定义一个测量函数$\mu_D : E \to V_e$：给定一个程序e，若程序e没有缺陷则$\mu_D(e) = A$ 否则 $\mu_D(e) = B$，即A表示程序没有缺陷，而B表示程序有缺陷。

测量映射

测量映射只考虑从E到V的映射，而没有考虑从经验关系系统中经验关系和经验操作到数值关系和数值操作之间的映射，范围过于宽泛，容易出现度量结果和人们经验理解不一致的地方，为此引入标度概念。

标度(Scale)

Definition (标度(Scale))

令$ERS = (E, R_1, \cdots, R_m, o_1, \cdots, o_n)$ 为一个经验关系系统，$NRS = (V, S_1, \cdots, S_m, \oplus_1, \cdots, \oplus_n)$ 为一个数值关系系统，$\mu : E \to V$ 是一个测量映射，则称(ERS, NRS, μ) 为一个标度，若对于经验关系系统的k元经验关系R_i和数值关系系统的k元经验关系$S_i (1 \leq i \leq m)$ 都有对于所有的$j (\in \{1, \cdots, n\})$ 和所有的$a_1, \cdots, a_k$，b，$c \in E$ 下式成立

$$R_i(a_1, \cdots, a_k) = S_i(\mu(a_1), \cdots, \mu(a_k))$$

以及

$$\mu(b\ o_j\ c) = \mu(b) \oplus_j \mu(c)$$

刻度：例子

- 程序经验关系系统ERS_P、程序长度数值关系系统NRS_{PL}和测量映射μ_L构成的$(ERS_P,\ NRS_{PL},\ \mu_L)$是一个标度，
- 这是因为给定两个程序$e1$ 和$e2$，若$e1 \gg e2$ 则$e2$ 是$e1$的子程序，从而$e1$的长度大于等于$e2$的长度，即$\mu_L(e1) \geq \mu_L(e2)$，
- 同时$e1;e2$的长度等于$e1$长度与$e2$长度之和，即$\mu_L(e1;e2) = \mu_L(e1) + \mu_L(e2)$。

刻度：例子

- 程序经验关系系统ERS_P、程序缺陷数值关系系统NRS_{PE}和测量映射μ_D构成的(ERS_P, NRS_{PE}, μ_D)也是一个标度，
- 这是因为给定两个程序$e1$和$e2$，若$e1 \gg e2$ 则$e2$ 是$e1$的子程序，若$e2$有缺陷则$e1$一定有缺陷，即若$\mu_D(e2) = B$ 则$\mu_D(e1) = B$，
- 同时即使$e2$没有缺陷也不能判断$e1$没有缺陷，即在$\mu_D(e2) = A$的前提下，$\mu_D(e1) = A$或B，因此有$\mu_D(e1) \sqsupseteq \mu_D(e2)$;
- 另外，若$e1$和$e2$中有一个有缺陷，则$e1; e2$ 一定有缺陷，进而得到$\mu_D(e1; e2) = \mu_D(e1) \wedge \mu_D(e2)$。

6.4 Extensive结构

Extensive结构

Extensive结构是一种特殊经验关系系统，由度量对象集A、度量对象上的二元关系R、度量对象上一个二元运算$\circ$三部分组成，其中

- 二元关系满足强完备性、传递性；
- 二元闭操作需满足弱结合律、单调性和阿基米德性。

物理科学中关于诸如长度、体积等的度量都是基于该结构。目前该结构也用于程序复杂性等内部属性的度量中.

Extensive结构

Definition (Extensive 结构)

一个经验关系系统$(A，R，\circ)$称为具有Extensive结构，若下列各条成立：

1. 弱序性：$(A，R)$ 是一个弱序，即R满足传递性和强完备性，强完备性是指对$\forall a，b \in A$，有$(a，b) \in R$或$(b，a) \in R$;
2. 弱结合律：$\forall a，b，c \in A$：$(a \circ b) \circ c \approx a \circ (b \circ c)$;
3. 单调性：$\forall a，b，c \in A$：$(a，b) \in R$ 当且仅当 $((a \circ c)，(b \circ c)) \in R$当且仅当 $((c \circ a)，(c \circ b)) \in R$;
4. 阿基米德性：如果$\forall a，b，c，d \in A$： 如果$(a，b) \in R_s$，则存在正整数 n 使得$((na \circ c)，(nb \circ d)) \in R_s$成立。

其中，

$$a \approx b \overset{\text{定义}}{=\!=} (a,b) \in R \text{ 并且 } (b,a) \in R$$

$$(a,b) \in R_s \overset{\text{定义}}{=\!=} (a,b) \in R \text{ 但是 } (b,a) \notin R$$

$1a = a$ 归纳定义 $(n+1)a = na \circ a$

Extensive结构表示和唯一性定理

Theorem (Extensive结构表示和唯一性定理)

设$(A, \succeq, \circ)$是一个经验关系系统，$\Re$是所有实数组成的集合。则存在(实值)测量函数$\mu: A \to \Re$ 使得对于任意$a, b \in A$使下式成立

$$a \succeq b \Leftrightarrow \mu(a) \geq \mu(b) \text{ 及 } \mu(a \circ b) = \mu(a) + \mu(b)$$

当且仅当$(A, \succeq, \circ)$是具有*Extensive* 结构。
如果另外一个(实值)测量函数μ'满足上式，则存在一个正实数r使得

$$\mu' = r\mu$$

成立。

Extensive结构表示和唯一性定理

- 这个定理说：具有Extenvie结构的经验关系系统一定有一个实值数值关系系统$(\Re, \geq, +)$，并且在实数倍的前提下是唯一的，即若有两个实值数值关系系统，则它们的测量函数相差实数倍。
- 我们经常接触到的物理科学中关于诸如长度、体积、时间等的度量都是基于Extensive结构。

Extensive结构表示和唯一性定理

- 程序经验关系系统NRS_P是具有Extensive结构，并且存在一个到程序长度数值关系系统$NRS_{PL} = (V_I, \geq, +)$的实值测量函数$\mu_L$ 将程序e的行数指定给程序e，即$\mu_L(e)$ =程序e的行数。
- 目前该结构应用于程序复杂性等内部属性度量中，但尚未被用于外部属性以及软件可信性度量中，主要原因是针对某一外部属性或者软件可信性难于构建满足弱序等一系列公理并被大家所接受的度量对象间经验关系，从而也就难于构建相应的Extensive结构。

6.5 面向源代码的软件可信性度量模型

6.5 面向源代码的软件可信性度量模型

本节选择基于Extensive结构进行软件可信性度量。

- 需建立用于刻画度量对象集、度量对象间经验关系和度量对象间操作的经验关系系统；
- 构造用于描述度量对象值集、值间关系以及值间操作的数值关系系统；
- 构造一个从经验关系系统到数值关系系统的测量函数，该函数把度量对象、对象间经验关系和对象间操作分别映射到数值关系系统中相对应的值、值间关系和值间操作。这个测量函数还应该是从经验关系系统到数值关系系统的同态映射。
- 本节接下来工作主要围绕这几个方面展开。

软件可信性经验关系系统

- 要对度量对象进行度量，首先必须给出它们的抽象表示，这样才可以对它们进行分析，对它们相关属性进行量化。本章是从源代码角度对软件可信性进行度量，因此下面所给出的抽象模型是关于源程序的抽象表示。
- 我们将整个程序看做一个系统，系统是由各个不同模块通过控制依赖或者数据依赖等关系连接在一起得到，而模块本身又是一个更小的系统，又有相应的程序元素并通过一定的关系进行相关联。
- 程序元素，简称为元素，包括变量、常量、表达式、各种类型语句和程序单元。
- 程序单元，简称单元，是一个实现特定目的可以被调用的代码，像C语言中的函数，Java语言中的方法等。

6.5 面向源代码的软件可信性度量模型

Definition (系统)

软件软件系统S为一个二元组(E, R)，其中E是S中程序元素组成的集合，R是E上一个二元关系，即$R \subseteq E \times E$。

Definition (模块)

给定一个系统$S = (E, R)$，称系统$m = (E_m, R_m)$为S的一个软件模块，简称模块，若

$$(E_m \subseteq E) \wedge (R_m \subseteq R)$$

成立。

比如E可以是程序中的语句或者程序块组成的集合，而R可以是这些语句或者程序块之间的控制流图，一个模块$m = (E_m, R_m)$可以是代码片段或子程序。
最好一个模块实现一个功能。即，按照功能划分模块。

C语言编写的快速排序程序QS

```c
#include <stdio.h>
int Partition(int *R,int i,int j)
{
    int pivot=R[i];
    while(i<j)
    {
        while(i<j&&R[j]>=pivot)
            j--;
        if(i<j)
            R[i++]=R[j];
        while(i<j&&R[i]<=pivot)
            i++;
        if(i<j)
            R[j--]=R[i];
    }
    R[i]=pivot;
    return i;
}
void QuickSort(int *R,int low,int high)
{
    int pivotpos;
    if(low<high)
    {
        pivotpos =Partition(R,low,high);
        QuickSort(R,low,pivotpos-1);
        QuickSort(R,pivotpos+1,high);
    }
}//QuickSort
int main()
{
    int s[100];
    printf("input␣10␣number");
    for(int i=0;i<10;i++)
    {
        scanf("%d",&s[i]);
    }
    QuickSort(s,0,9);
    for (int j=0;j<10;j++)
    {
        printf("%d␣",s[j]);
    }
}
```

给定一个程序系统$S=(E，R)$，令MS表示系统S中所有模块组成的集合，则MS为我们所要构建的经验关系系统的度量对象集。

Definition (模块二元运算$\circ_T$)

设$m_1=(E_{m_1}，R_{m_1}), m_2=(E_{m_2}，R_{m_2})\in MS$，定义模块$E_{m_1}$和$E_{m_2}$间的二元关系$R_{m_1m_2}$:

$$R_{m_1m_2}=\{(e_1，e_2)\mid e_1\in E_{m_1}\wedge e_2\in E_{m_2}\wedge(e_1，e_2)\in R\}$$

则两模块间的二元操作$\circ_T$定义如下

$$m_1\circ_T m_2=(E_{m_1}\uplus E_{m_2}，R_{m_1}\cup R_{m_2}\cup R_{m_1m_2})$$

其中，$E_{m_1}\uplus E_{m_2}$表示由E_{m_1}和E_{m_2}中所有元素组成的多重集，即若程序单元e在两个模块E_{m_1}出现n_1次在模块E_{m_2}中出现n_2次，则在e在模块$E_{m_1}\uplus E_{m_2}$出现n_1+n_2次。这样程序单元在一个模块中可能出现多次。由模块定义得$m_1\circ_T m_2\in MS$，即$\circ_T$是模块间的二元闭操作，即二元运算。

C语言编写的快速排序程序QS

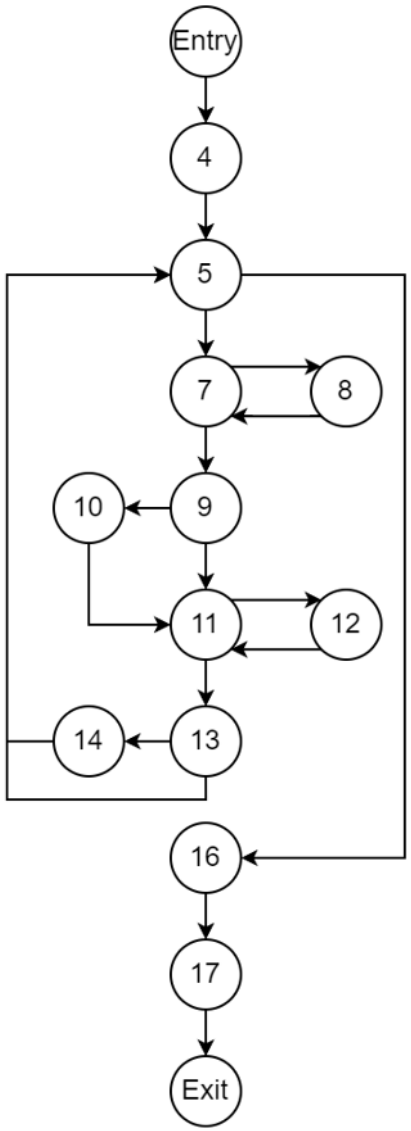
```c
#include <stdio.h>
int Partition(int *R,int i,int j)
{
    int pivot=R[i];
    while(i<j)
    {
        while(i<j&&R[j]>=pivot)
            j--;
        if(i<j)
            R[i++]=R[j];
        while(i<j&&R[i]<=pivot)
            i++;
        if(i<j)
            R[j--]=R[i];
    }
    R[i]=pivot;
    return i;
}
void QuickSort(int *R,int low,int high)
{
    int pivotpos;
    if(low<high)
    {
        pivotpos =Partition(R,low,high);
        QuickSort(R,low,pivotpos-1);
        QuickSort(R,pivotpos+1,high);
    }
}//QuickSort
int main()
{
    int s[100];
    printf("input␣10␣number");
    for(int i=0;i<10;i++)
    {
        scanf("%d",&s[i]);
    }
    QuickSort(s,0,9);
    for (int j=0;j<10;j++)
    {
        printf("%d␣",s[j]);
    }
}
```

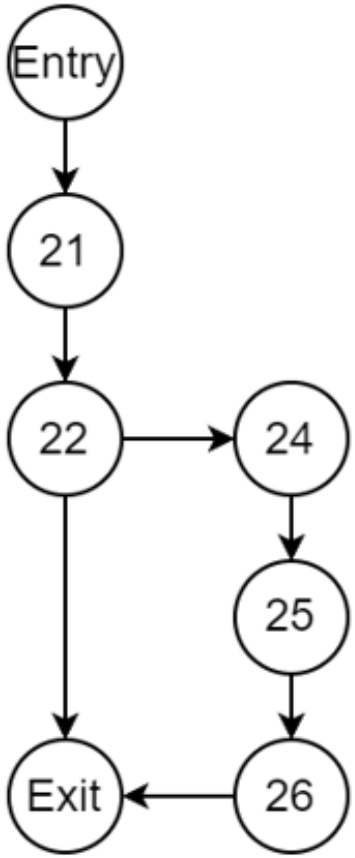
C语言编写的快速排序程序QS

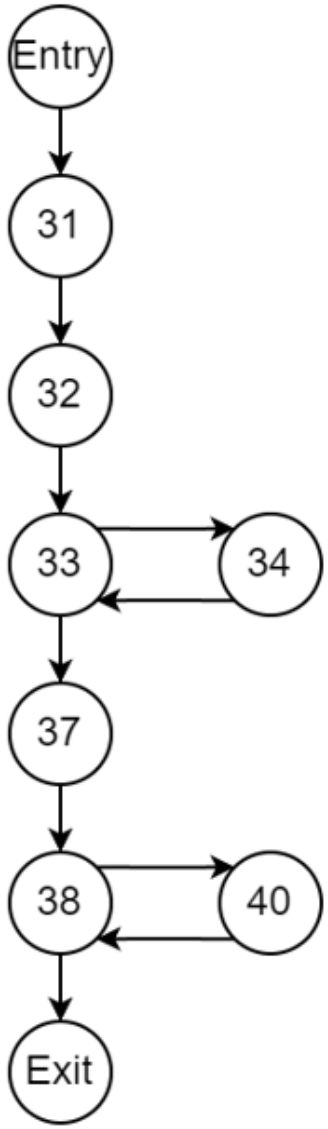
这个软件含有42行代码，第一行是头文件，后面各模块都要使用它，因而不记载模块中。
将这个代码分成三个模块：令$M = \{m_1 =< E_{m_1}, R_{m_1} >, m_2 =< E_{m_2}, R_{m_2} >, m_3 =< E_{m_3}, R_{m_3} >\}$，其中

- E_{m_1}由第2-18行中语句组成的集合，
- E_{m_2} 由第19-28行中语句组成的集合，
- E_{m_3} 由第29-42行中语句组成的集合。

C语言快速排序程序QS:模块M1、M2、M3的控制流图







6.5 面向源代码的软件可信性度量模型

- 软件可信性是软件行为和结果符合用户预期的能力，无论是软件行为还是运行结果都由其实现机制所决定。
- 因此，给定一个软件，其可信性可通过求解软件本身所采用实现机制和用户期望所采用实现机制的相似程度来建立；
- 同时软件本身又是由模块通过控制或数据依赖等关系组成的集合，这样软件可信性度量又可转化为其模块所采用实现机制和用户期望模块所采用实现机制的相似程度来计算。
- 另一方面，模块是由程序元素依据一定关系组成的集合，从而软件可信性度量又可进一步通过计算程序元素所采用实现机制和用户期望程序元素所采用实现机制的相似程度来得到。

6.5 面向源代码的软件可信性度量模型

- 给定一个程序系统$S=(E,R)$, MS是S的模块集合，$E_m \in MS$。
- 符号$\#(E_m)_{expect}$表示观察到模块m中具有用户期望机制实现程序元素个数，$\#(E)$ 表示E 中所有程序元素个数。
- 先引入符号$\kappa(m)$ 表示观察到模块$m=(E_m,R_m)$中具有用户期望机制实现程序元素个数与E中所有程序元素个数的比值，即

$$\kappa(m)=\frac{\#(E_m)_{expect}}{\#(E)}$$

Claim

给定两个模块$m_1=(E_{m_1},\ R_{m_1})$，$m_2=(E_{m_2},\ R_{m_2}) \in MS$，则$\kappa(m_1 \circ_T m_2)=\kappa(m_1)+\kappa(m_2)$。

按照模块二元运算定义进行证明。

C语言编写的快速排序程序QS

```c
#include <stdio.h>
int Partition(int *R,int i,int j)
{
    int pivot=R[i];
    while(i<j)
    {
        while(i<j&&R[j]>=pivot)
            j--;
        if(i<j)
            R[i++]=R[j];
        while(i<j&&R[i]<=pivot)
            i++;
        if(i<j)
            R[j--]=R[i];
    }
    R[i]=pivot;
    return i;
}
void QuickSort(int *R,int low,int high)
{
    int pivotpos;
    if(low<high)
    {
        pivotpos =Partition(R,low,high);
        QuickSort(R,low,pivotpos-1);
        QuickSort(R,pivotpos+1,high);
    }
}//QuickSort
int main()
{
    int s[100];
    printf("input␣10␣number");
    for(int i=0;i<10;i++)
    {
        scanf("%d",&s[i]);
    }
    QuickSort(s,0,9);
    for (int j=0;j<10;j++)
    {
        printf("%d␣",s[j]);
    }
}
```

C语言编写的快速排序程序QS

我们只考虑变量程序元素。

将这个代码分成三个模块：令$M = \{m_1 =< E_{m_1}, R_{m_1} >, m_2 =< E_{m_2}, R_{m_2} >, m_3 =< E_{m_3}, R_{m_3} >\}$，其中

- E_{m_1}由第2-18行中语句组成的集合，$E_{m_1} = \{R, i, j, pivot\}$;
- E_{m_2} 由第19-28行中语句组成的集合，$E_{m_2} = \{R, low, high, pivotpos\}$;
- E_{m_3} 由第29-42行中语句组成的集合,$E_{m_3} = \{s[100], i, j\}$。
- $\#(E_{QS}) = 11$（？）。

Claim

变量$i, j \in E_{m_1}$不止一个用途，同样标识符在$main$函数也被使用，所以它们不是用户所期望的。此外，$R \in E_{m_1}$ 和$R \in E_{m_2}$ 不具有自描述性，所以也不是用户所期望的。这样可以得到$\#(E_{m_1})_{expect} = 1$，$\#(E_{m_2})_{expect} = 3$，$\#(E_{m_3})_{expect} = 1$，$\#(E_{QS}) = 11$。从而有$\kappa(m_1) = \frac{1}{11}$，$\kappa(m_2) = \frac{3}{11}$，$\kappa(m_3) = \frac{1}{11}$。

6.5 面向源代码的软件可信性度量模型

Definition (模块间二元经验关系$\succsim_T$)

设$m_1 = (E_{m_1}, R_{m_1})$，$m_2 = (E_{m_2}, R_{m_2}) \in MS$，则模块$m_1$可信性大于等于模块$m_2$可信性的经验关系，记作$m_1 \succsim_T m_2$，当且仅当$\kappa(m_1) \geq \kappa(m_2)$。

Definition (软件模块可信性经验关系系统)

给定一个程序系统$S = (E, R)$，称$(MS, \succsim_T, \circ_T)$为一个软件模块可信性经验关系系统。

6.5 面向源代码的软件可信性度量模型

Theorem

给定一个程序系统$S=(E, R)$，令MS为该系统中所有模块组成的集合，$\Re^+$为正实数集，$m=(E_m, R_m)\in MS$。则函数κ是模块可信性经验关系系统$(MS, \succsim_T, \circ_T)$的实值测量函数，即保持二元闭运算$\circ_T$和二元经验关系$\succsim_T$。

由Extensive结构表示和唯一性定理得到下面结论

Theorem

软件模块可信性经验关系系统$(MS, \succsim_T, \circ_T)$具有*Extensive*结构。

Extensive结构表示定理保证存在从Extensive结构到某数值关系系统具有比率标度的可加函数，并且反之亦然。下面就依据已建立的模块可信性经验关系系统$(MS, \succsim_T, \circ_T)$建立软件模块可信性度量模型。

软件模块可信性经验关系系统的Extensive结构性

Proof.

按照Extensive 结构的定义，我们需要验证4条基本性质：弱序性、弱结合律、单调性和阿基米德性。下面我们逐条进行验证。
首先定义关系$\approx$:
给定两个模块m_1和m_2，$m_1 \approx m_2$ 当且仅当$m_1 \succsim_T m_2$ 与$m_2 \succsim_T m_1$都成立。由关系$\succsim_T$定义可得：$m_1 \approx m_2$当且仅当$\kappa(m_1) = \kappa(m_2)$。
(1) $\succsim_T$是弱序的：验证$\succsim_T$是传递的和强完备的。
(a) 传递性:
(b) 强完备性：
(2) 弱结合律：因为$m_1 \circ_T m_2 = (E_{m_1} \uplus E_{m_2},\ R_{m_1} \cup R_{m_2} \cup R_{m_1 m_2})$，所以

$$(m_1 \circ_T m_2) \circ_T m_3 = ((E_{m_1} \uplus E_{m_2}) \uplus E_{m_2},$$
$$R_{m_1} \cup R_{m_2} \cup R_{m_3} \cup R_{m_1 m_2} \cup R_{m_1 m_3} \cup R_{m_2 m_3} \cup R_{m_1 m_2 m_3})$$

软件模块可信性经验关系系统的Extensive结构性

Proof.

同理，因为$m_2 \circ_T m_3 = (E_{m_2} \uplus E_{m_3},\ R_{m_2} \cup R_{m_3} \cup R_{m_2m_3})$ 所以

$$m_1 \circ_T (m_2 \circ_T m_3) = (E_{m_1} \uplus (E_{m_2} \uplus E_{m_2}),$$
$$R_{m_1} \cup R_{m_2} \cup R_{m_3} \cup R_{m_1m_2} \cup R_{m_1m_3} \cup R_{m_2m_3} \cup R_{m_1m_2m_3})$$

由$\uplus$定义可知$\uplus$满足结合律，且$\cup$满足结合律，所以有

$$(m_1 \circ_T m_2) \circ_T m_3 \approx m_1 \circ_T (m_2 \circ_T m_3)$$

成立。

(3) 单调性：首先证明$\forall m_1$，m_2，$m_3 \in MS$，若$m_1 \succsim_T m_2$，则$m_1 \circ_T m_3 \succsim_T m_2 \circ_T m_3$。

由$m_1 \succsim_T m_2$得$\kappa(m_1) \geq \kappa(m_2)$，所以有

$$\kappa(m_1 \circ_T m_3) = \kappa(m_1) + \kappa(m_3) \geq \kappa(m_2) + \kappa(m_3) = \kappa(m_2 \circ_T m_3)$$

软件模块可信性经验关系系统的Extensive结构性

Proof.

这样有$m_1 \circ_T m_3 \succsim_T m_2 \circ_T m_3$。
接下来证明$\forall m_1$，m_2，$m_3 \in MS$，若$m_1 \circ_T m_3 \succsim_T m_2 \circ_T m_3$，则$m_1 \succsim_T m_2$。
(4) 阿基米德性：任取MS中的模块m_1，m_2，m_3，m_4，再设$m_1 \succsim_T m_2$且$m_2 \not\succsim_T m_1$，则$\kappa(m_1) \geq \kappa(m_2)$且$\kappa(m_1) \neq \kappa(m_2)$。
取正整数n使得$n \geq \dfrac{\kappa(m_4) - \kappa(m_3)}{\kappa(m_1) - \kappa(m_2)}$成立。进而有

$$n\kappa(m_1) + \kappa(m_3) \geq n\kappa(m_2) + \kappa(m_4)$$

成立。这样有

$$nm_1 \circ_T m_3 \succsim_T nm_2 \circ_T m_4$$

成立。
综上所述$(MS, \succsim_T, \circ_T)$具有Extensive结构。 □

作业一

详细证明：软件模块可信性经验关系系统$(MS, \succsim_T, \circ_T)$具有Extensive结构。