Data structure

**//状态机状态节点**

St\_node {

Id , // 节点Id，即在状态转移矩阵中的下角标

S // 一个n维向量 (s1,…,sn）每个元素 si 表示第i个承诺目前的状态值

};

**//状态转移矩阵**

M=[m][m] // m\*m 的矩阵，其中m表示的是状态机的状态节点个数 M(i,j)有如下定义

// M(i,j) = 空 表示第i个st不能转移到 第j个st

// M(i,j) != 空 表示第i个st在接收到该事件后转移到 第j个st

**//承诺*cc***

cc { x,y,pre,res,tc}; // 承诺 是五元组 分别为（承诺人，被承诺人，前提，结果，时间）

//cc 有5种状态 1 ：激活;2：就绪;3：满足;4：过期;5：违约

List CC; // 用于记录n 个承诺的信息列表 cc的数组

List queue; //储存状态机节点的队列

queue.append(st0); // 初始状态节点入队

while (len(queue)>0){ //当状态队列长度不为0 就遍历合理状态

List Repeat <- *Φ* //检查是否有重复 将所有新生成的状态且队列中没有的入队

st <- queue.pop(); // 节点出队

( children,Repeat) <- createChildrenNodes(CC,st,Repeat);

// 返回所有新的合理子节点以及查重列表

for (each (st’ in children){ //将所有合理子状态节点入队

queue.append(st’);

}

}

//输入状态节点状态 节点st，重复列表Repeat，以及输入所有承诺信息

//返回所有合理的子状态节点以及重复列表，并将状态转移记录到M中

createChildrenNodes( CC,st,Repeat){

s<-st.S //取出该节点中的状态

List rlt <- *Φ* ; //建立列表存储所有的子节点

for(i=1:n){ //遍历状态中的所有承诺的状态 跟据承诺的定义只有1 （激活）2（就绪） 才能转变

if (s[i]==1） //当此承诺状态为1时

if（cc in CC[i].pre） //如果承诺以其他承诺为前提

if sat //如果前提条件中相应承诺满足 则置为2

change = 2 ; // 当承诺的前提按时完成时 则置为2

(rlt,Repeat)= handel（st,i,change,timeout(CC[i]).pre,Repeat）;

// 调用处理函数

else //如果前提不满足

change = 4 ; // 当承诺的前提过期时 则置为4

(rlt,Repeat)= handel（st,i,change,timeout(CC[i]).pre,Repeat）;

// 调用处理函数 返回子节点矩阵与重复矩阵

else //如果承诺前提中不依赖其他承诺 则可直接置为2

change = 2 ;

(rlt,Repeat)= handel（st,i,change,timeout(CC[i]).pre,Repeat）; // 调用处理函数

change = 4 ; // 当承诺的前提过期时 则置为4

(rlt,Repeat)= handel（st,i,change,timeout(CC[i]).pre,Repeat）;

// 调用处理函数 返回子节点矩阵与重复矩阵

if (s[i]==2）

change = 3 ; // 当承诺的结果按时完成时 则置为3

(rlt,Repeat)= handel（st,i,change,ture(CC[i]).res,Repeat）; // 调用处理函数

change = 5; // 当承诺的结果过期时 则置为5

(rlt,Repeat)= handel（st,i,change,timeout(CC[i]).pre,Repeat）; // 调用处理函数

}

return （rlt,Repeat）; //返回子的节点队列

}

// 处理状态转移函数 生成新状态，并将新生成的状态加入状态队列，并存入转移矩阵

// 返回子状态的列表,重复矩阵

handel (st,i,change,event,rlt,Repeat){

s<-st.S //取出该节点中的状态

new\_s=s ; // 生成成新的状态

new\_s[i] = change; // 状态改变

if (new\_s in Repeat） // 检查是否重复 如果新的状态已经在状态队列中;

st’.Id = 对应的节点的id ; // 若存在则取出对应的id

else

st’=new St\_node(,new\_s); // 若不存在，创建新的状态机节点

Repeat.add(st’); // 如不存在则将新生成状态入队

rlt.add(st’); // 并且将新生成的节点加入到子节点队列

M（st.Id，st’.Id） = event ; // 存入转移矩阵

return (rlt,Repeat); // 返回新的子状态列表以及重复列表

}