# 软件静态结构的依赖网络建模方法与特性分析*)

钱冠群 张 莉 张 林 Philip Lew
(北京航空航天大学软件工程研究所 北京 100083)

**摘 要** 现有的软件复杂网络建模方法,对模型的定义各不相同,缺少统一的框架,导致不同编程语言、不同粒度的软件模型分析对比困难。针对软件的静态结构,分析对比了现有建模方法的特点,提出了软件依赖网络建模方法,将软件系统抽象为计算服务提供者以及它们之间的依赖关系,实现了不同粒度下网络模型的直观映射;通过对 Java 软件依赖网络的实例分析,证明其具有小世界、高聚类系数的特性,入度基本服从幂率分布,出度基本服从带指数截断的幂率分布,并指出这些网络特性的软件工程含义,探讨了它们对软件开发和维护的指导意义。

**关键词** 软件工程,复杂网络,软件维护

## Modeling Method and Characteristics Analysis of Software Dependency Networks

QIAN Guan-qun ZHANG Li ZHANG Lin Philip Lew
(Software Engineering Institute, Beihang University, Beijing 100083, China)

**Abstract** In previous study of software complex network modeling, the lack of unified definition of model makes it difficult to compare between software networks from different program languages or abstraction levels. Analyzed the current methods and proposed software dependency network method. In this method, software system is abstracted as service providers and dependencies between them. Mapping can be easily achieved between networks from different abstract levels. Following the method, study average degree, degree distribution, average distance and clustering coefficient of several Java software packages. It is approved that software dependency networks display small world, high clustering coefficient. In-degree follows power law and out-degree follows power law with exponent cutoff. The underlying principles and guidance to software development and maintenance are also discussed.

**Keywords** Software engineering, Complex networks, Software maintenance

## 1 引言

随着软件规模的不断扩大,软件是一种复杂系统的观点越来越被人们所接受。

复杂网络作为大量真实复杂系统的拓扑抽象,已成为理解复杂系统的有效手段。使用网络的观点来看待软件,运用复杂网络理论,定量研究软件的拓扑结构,并由此分析软件拓扑结构的特征属性及其演化机理,为软件的开发和维护过程提供理论依据,已经逐渐成为软件工程研究的一个热点[1-8]。但是,当前软件的复杂网络研究缺少一致的模型定义,这给网络的特征分析带来了不便。

本文从软件静态结构的网络建模方法出发,首先分析对比现有建模方法的差异和不足,在此基础上提出了软件依赖网络的模型定义。其次,根据这个定义,对 8 个 Java 软件包的网络特征(度分布、平均距离和聚类系数)进行了分析和对比,并讨论这些特性的软件工程含义,以及对软件开发和维护的指导意义。

## 2 软件静态结构的网络建模方法

使用复杂网络理论研究软件系统的拓扑结构,首要的问题是如何定义软件系统的网络模型,以及如何从软件制品中获得对应的模型。

### 2.1 现有的网络建模方法分析

#### 2.1.1 网络模型定义

长期以来,图一直是软件系统进行建模、分析和度量的有效手段。从结构化软件系统分析中使用的数据流图、控制流图,到当前面向对象、分布式系统建模中广泛使用的 UML,都是以图为基础。现有的软件静态结构的复杂网络建模方法,借鉴了传统软件工程的建模方法,主要针对结构化和面向对象软件。

其中,最直观的方法就是将不同软件的编程元素(子程序、类等)以及它们之间关系构造成软件系统的基础拓扑结构。结构化软件以调用关系图为基础,将子程序或函数作为节点,子程序或函数之间的调用关系映射为有向或无向边[2]。面向对象软件则以类图为基础,将类作为节点,类间关系映射为有向或无向边[2,3,5,9],所不同的是有的只选取类间主要的继承和组成关系[2,3,5],有的则包含继承、依赖、关联、聚合和组成关系[9]。

在基于编程元素的建模方法基础上,研究人员还提出了文件级和构件级(例如包)[7]的软件复杂网络建模方法。将文件或包作为节点,文件或包间的包含或依赖关系作为边。这类方法可以看作是前一类方法的一种延伸,而由这些方法得

*)本文得到国家自然科学基金项目"软件适应性预测模型的研究"(60773155)的资助。钱冠群 博士生,主要研究领域为软件体系结构;张 莉 教授,博士生导师,主要研究领域为软件体系结构、企事业过程建模;张 林 博士生,主要研究领域为软件体系结构;Philip Lew 博士留学生,主要研究领域为软件体系结构。

到的软件网络可以看作是对基于编程元素建模得到的软件网络的一种抽象，只是省略了文件或构件内部的网络细节。

另外，De M A 等人还提出一种基于协作关系的网络建模方法[8]，其思想类似于演员和科学协作网[10,11]。这种方法以 C/C++的头文件作为节点，出现在同一个 C/C++源文件中的头文件之间用无向边相连。这种方法本质上是对文件间包含或依赖网络的一种变换。

根据上面的分析，我们认为：编程元素级的软件网络建模方法是其它大粒度软件网络建模方法的基础；其次，有必要建立编程元素级、文件级、构件级等不同粒度的软件网络模型之间的对应关系，为软件的开发和维护提供帮助。

以下我们着重讨论编程元素级的软件网络建模方法。

### 2.1.2 网络模型的抽取方法

网络模型的抽取方法，是指根据软件系统的复杂网络建模需要，从软件制品中获取对应的软件网络的过程，它的准确度和效率直接影响后续的定量分析过程。

编程元素级的网络模型抽取方法可以分为 3 种：一种是扫描源代码，根据语法关键字的分析过滤，得到软件的网络模型[3]；另一种是扫描编译后的中间结果，例如 Java 语言的 .class或.jar 文件、GCC 的抽象语法树文件，通过解析这些中间结果来获得软件的网络模型[5]；第三种是通过简化 UML 类图来获得软件的网络模型，在缺少设计模型时，可以先通过 Case 工具逆向工程得到软件的 UML 类图，然后进一步得到软件的网络模型[9]。

### 2.1.3 现有方法的建模结果对比

现有的国内外研究对结构化软件的网络模型定义比较统一，但在面向对象软件上区别很大。下面我们通过一个具体实例，对典型的面向对象软件的网络建模方法做对比分析。

```
public class v1 {
int x;
};
public class v2 extends v1 {
void m2() {
}
}
public class v3 extends v1 {
void m3(v1 _a,v2 _b) {
}
}
public class v4 {
void m4(v2 _a) {
}
};
```

图 1 Java 软件实例

表 1 是不同建模方法对图 1 中 Java 代码的建模结果对比。通过实验和结果分析，我们发现：

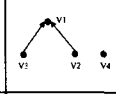
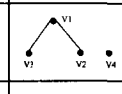
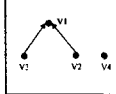
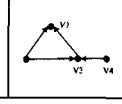
①使用有向图，可以更好表达软件系统的拓扑结构。文献[3]和文献[5]对网络的定义相似，对实例代码的建模结果也很相似。但无向网络不区分连接的出入关系，会造成结构信息的丢失，不利于后续的网络特性分析[2]。

②采用简化 UML 类图抽取软件网络的建模方法，很大程度上依赖于 UML 类图信息的准确性。如果使用 Case 工具逆向工程获得 UML 类图，Case 工具逆向工程的准确性和速度直接影响到该方法的有效性。

③面向对象软件网络建模时，应该尽可能保留类图中定义的所有关系类型，提高后续分析的有效性。表中的 M4 模型由于没有定义 import 关系，相对 M5 模型，会丢失一些有效信息。

④如果将 v1，v2，v3 封装成构件 A，v4 封装成构件 B，M5 模型通过图化简，可以直接得到构件 B 依赖于构件 A，其它模型无法得到相同的结果。这说明，M5 模型与大粒度软件网络模型之间存在直观的映射关系，更适合做软件的体系结构分析。

表 1 不同网络建模方法的结果对比

| | 网络类型 | 模型抽取方法 | 软件网络模型 | 模型编号 | 备注 |
|---|---|---|---|---|---|
| 文献[3] | 有向图 | 扫描源代码 |  | M1 | 只提取继承或组成关系 |
| 文献[5] | 无向图 | 扫描中间结果 |  | M2 | 只提取继承或组成关系 |
| 文献[9] | 有向图 | 逆向工程＋简化 UML 类图 |  | M3 | 使用 EclipseUML 逆向工程得到的 UML 类图 |
| | | |  | M4 | 由类图化简得到的软件网络模型——不考虑 import 关系 |
| | | |  | M5 | 由类图化简得到的软件网络模型——考虑 import 关系 |

### 2.1.4 现有方法的不足

通过上面的分析，我们认为现有的软件网络建模方法存在以下的不足：

①现有的编程元素级的网络建模方法不能同时适用于不同编程语言、不同粒度的软件网络建模需要。例如，结构化软件的建模方法就不能适用于面向对象软件的网络建模，编程元素级的建模方法又不一定适合构件级软件的建模。

②现有的编程元素级的网络建模方法，由于对网络中点和线的定义各不相同，使得后续的网络特性分析，特别是定量分析的结论存在很大的差异，这给不同编程语言、不同粒度的软件模型的特征分析和对比带来了困难。

③当前的不同粒度的软件网络模型缺乏一致的定义方式，无法实现不同粒度模型之间直观的映射关系。

因此亟需一种统一的软件网络模型定义，能够适用于不同编程语言、不同粒度的软件网络建模，并能实现同一软件的不同粒度网络模型之间的直观映射，为软件的网络建模和分析提供依据。

## 2.2 软件依赖网络的定义

软件系统从根本上看，都可以抽象为计算服务提供者以及它们之间的依赖关系。在结构化软件中，服务提供者是函数，服务提供者之间的依赖关系表现为函数之间的调用关系。例如，函数 A 调用子函数 B，说明 A 依赖于 B 所提供的服务。在面向对象软件中，服务提供者是类或接口，类或接口间的继承、依赖、关联、聚合和组成等关系在本质上也是一种程序调用关系即服务之间的依赖关系。例如类 A 继承类 B，说明 A 将使用 B 中的接口和实现，即 A 依赖于 B 提供的服务。

这样，就从概念上统一了不同类型，不同粒度的软件网络模型定义，同时实现了不同粒度模型之间的直接映射关系。

我们将这种抽象模型称作软件的依赖网络模型。

以下给出软件依赖网络的形式化定义。

**定义 1** 任何一个软件都可以表示为一个软件依赖网络 $G=(V,E)$。其中，$G$ 是有向无权网络；$V$ 是节点的集合，$V$ 中的每个元素 $v_i$ 对应软件中的一个计算服务提供者；$E$ 是边的集合，$E$ 中的每个元素 $\langle v_i,v_j\rangle$ 是一个有序对，当且仅当 $v_i$ 使用了 $v_j$ 提供的服务时，$\langle v_i,v_j\rangle\in E$。

根据软件依赖网络的定义，可以将面向对象软件的编程元素级依赖网络定义如下。

**定义 2** 任何一个面向对象软件，其编程元素级网络模型可以表示为 $G=(V,E)$。其中，$G$ 是有向无权网络；$V$ 是节点的集合，$V$ 中的每个元素 $V_i$ 对应软件中的一个类/接口；$E$ 是边的集合，$E$ 中的每个元素 $\langle v_i,v_j\rangle$ 是一个有序对，当且仅当 $v_i$ 使用了 $v_j$ 提供的服务时，$\langle v_i,v_j\rangle\in E$。

根据定义 2，我们可以得到图 1 中 Java 代码对应的网络模型，参见图 2。该模型与表 1 中的 M5 模型相同。

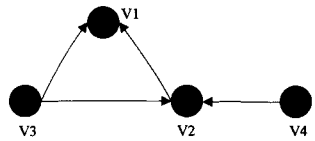



图 2 使用软件依赖网络定义得到的网络模型

表 2 8 个 Java 软件包的统计数据

| 软件库名称 | 大小 (KB) | 节点数 N=\|V\| | 边数 L=\|E\| | ⟨k⟩ | $C_{rand}$ | C | d | $d_{rand}$ |
|---|---|---|---|---|---|---|---|---|
| Apache antl. 6. 5 ant. jar | 1002 | 576 | 2431 | 4. 220 | 0. 0073 | 0. 256 | 4. 799 | 4. 414 |
| Apache cactusl. 7. 2. cactus-1. 7. 2. jar | 256 | 112 | 354 | 3. 161 | 0. 0282 | 0. 204 | 3. 028 | 4. 100 |
| Apache derby corel0. 1. 2 derby. jar | 2082 | 1183 | 11128 | 9. 407 | 0. 0080 | 0. 241 | 6. 346 | 3. 157 |
| Java3D 1. 5. 0. j3dcore. jar | 2841 | 495 | 4160 | 8. 404 | 0. 0170 | 0. 296 | 3. 512 | 2. 915 |
| org. eclipse. jdt. ui_3. 2. 1. r321_v20060907. jar | 7893 | 3871 | 16155 | 4. 173 | 0. 0011 | 0. 188 | 8. 236 | 5. 782 |
| jdkl. 5. 0_04 rt. jar | 36840 | 12689 | 125723 | 9. 908 | 0. 0008 | 0. 305 | 6. 630 | 4. 120 |
| jdkl. 5. 0_04 tools. jar | 6865 | 1912 | 12458 | 6. 516 | 0. 0034 | 0. 280 | 3. 774 | 4. 032 |
| Apache xerces2. 8. 0 xercesImpl. jar | 1174 | 878 | 4782 | 5. 446 | 0. 0062 | 0. 237 | 4. 854 | 3. 999 |

## 3 软件依赖网络的特性分析

本节以 8 个 Java 软件为例，分析面向对象软件的编程元素级依赖网络的特征。

在实例分析时，为了提高网络模型的抽取和分析效率，及时响应软件变更带来的结构变化，本文采用如下的模型抽取方法：首先使用面向 Java 的字节码依赖关系分析工具 DependencyFinder[12]，通过扫描. jar 文件，获取软件内部各个包、类/接口和特征(feature)之间的依赖关系。然后通过编程的方式，将其转换为类/接口间的依赖关系，进而得到软件的依赖网络。最后使用网络分析和可视化软件 pajek[13]，结合编程，对获得的网络进行统计特征分析。分析中，我们只考虑软件包内部各节点间的依赖关系，不考虑对第三方软件包的依赖。表 2 是 8 个软件包的统计数据对比。

### 3.1 节点数与边数

在复杂网络中，节点和边的数量反映了网络的规模和复杂程度。

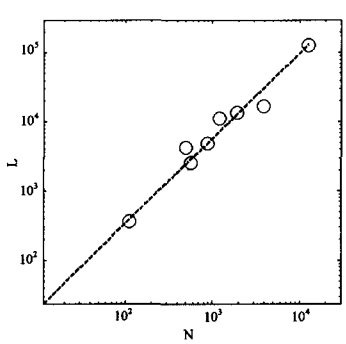



图 3 双对数坐标下 8 个软件的节点数与边数分布

图 3 是 8 个软件包的节点数 $N=|V|$ 和边数 $L=|E|$ 之间的对应关系。可以看到，在双对数坐标下，它们之间基本符合线性关系 $L\sim N^{1.224}$。这说明软件的依赖网络十分稀疏。换句话说，每向软件中增加一个类，平均只会与软件中已有的常数个类产生依赖关系。

### 3.2 度的分析

度是复杂网络节点的属性中最简单但也是最重要的性质。一个节点 $v_i$ 的度 $k_i$ 定义为与它连接的其它节点的数目。

有向网络中，一个节点的度又分为入度和出度。节点 $v_i$ 的出度是指从 $v_i$ 指向其它节点 $v_j$ 的边的数目，节点 $v_i$ 的入度是指从其它节点 $v_j$ 指向节点 $v_i$ 的边的数目。网络的节点平均度 $\langle k\rangle$ 定义为网络中所有节点 $v_i$ 的度 $k_i$ 的平均值。直观上来看，一个节点的度越大，就意味着这个节点越重要。

在软件依赖网络中，出度表示该节点所依赖的服务提供者数量，入度表示该节点被依赖的其它服务的数目。

#### 3.2.1 度分布

网络中所有节点的度的分布情况可以用分布函数 $P(k)$ 来描述。$P(k)$ 表示的是一个随机选定的节点的度恰好为 $k$ 的概率。为了方便对比，本文使用频数 $F(k)$ 来描述网络中所有节点的出入度分布情况，$F(k)$ 表示网络中度恰好为 $k$ 的节点的个数。

##### 3.2.1.1 入度分布

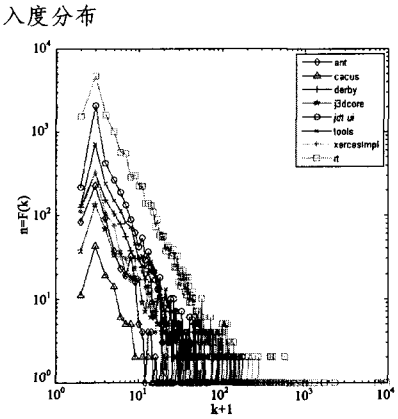



图 4 双对数坐标下 8 个软件包的入度分布情况

图 4 是双对数坐标系下，8 个软件包的入度分布情况，可以看到：

①虽然 8 个软件包的规模差异悬殊，它们的依赖网络的入度分布图形却十分相似；

②在 $k=1$ 时，节点个数出现峰值；

③当 $k\geqslant 1$ 时，节点个数基本符合幂率分布；

④在 $k=0$ 处，不符合 $k\geqslant 1$ 时的幂率分布。

3.2.1.2 出度分布

图 5 是双对数坐标系下 8 个软件包的出度分布情况，可以看到：

①8 个软件包的出度分布图形十分相似；

②在 $k=1$ 或 2 时，节点个数出现峰值；

③当 $k\geqslant 2$ 时，节点个数逐步递减，且符合带指数截断的幂率分布，即 $F(k)\sim k^{-r}e^{-k/\kappa}$；

④在 $k=0\sim 2$ 处，不符合 $k\geqslant 2$ 时带指数截断的幂率分布；

另外，从图中可以看到，rt. jar 中不存在 $k=0$ 的点。但在 Java 中，如果一个类或者接口不直接继承于其它父类，编译时会默认将该类继承于 java. lang. Object 类。由此可以推断，rt. jar 中出度为 0 的节点只可能是 java. lang. Object。分析 jdk 源代码后，我们发现：rt. jar 中类之间确实存在循环依赖的现象。例如：java. lang. String 继承于 java. lang. Object，同时 java. lang. Object 的实现也依赖于 java. lang. String。

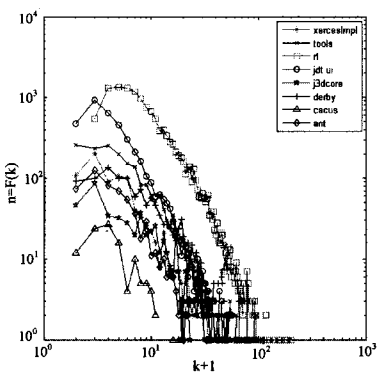



图 5 双对数坐标下 8 个软件包的出度分布情况

以下针对度分布来分析潜在的软件工程含义。

①如果节点入度较高，意味着这些节点对应的服务被高度复用，可能是软件中较核心或较基础的部分。改善这些服务的质量，例如可靠性、响应速度，可以大大改善整个软件的质量。因此，这些节点将是软件测试、程序理解以及后期维护的重点。

②如果节点出度较高，意味着该节点在提供自身计算服务时，需要组合其它多种服务。过大的软件出度意味着这些节点与外界的交互比较复杂，维护起来可能会比较困难。

③如果节点的出度为 0，表示该节点提供的是原子服务。

④如果节点的入度为 0，则表示该节点是软件中某个特定功能的入口。以 ant. jar 的依赖网络为例，共有 84 个入度为 0 的节点，其中 59 个来自 org. apache. tools. ant. taskdefs 包，每个节点都对应 ant 的一类任务。使用逐步分层的方式，可以快速定位软件各子功能的入口，明确各服务提供者之间的依赖关系，大大改善软件维护人员程序理解的效率，从而降低维护成本。

⑤如果节点的出入度都为 1，说明这些节点充当了相邻服务节点的桥梁。如果不考虑软件的其它质量属性，可以尝试将这些节点对应的服务转移到相邻的节点中，从而简化软件结构，提高信息的传递效率。

3.2.1.3 平均度分析

图 6 是 8 个软件包网络模型的节点个数与平均出度/入度对比关系。可以看到，随着软件依赖网络规模的增加，网络的节点平均度变化不大，大约在 4～10 之间。但是从图 4 和图 5 可以看到，节点的出入度的变化范围却很大，这说明软件依赖网络具有明显的异质性。

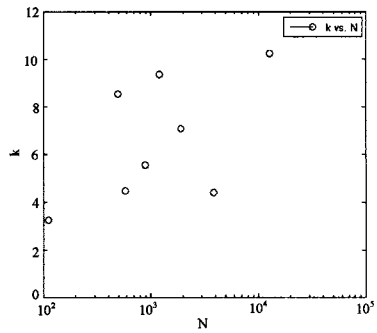



图 6 8 个软件包的平均度

3.2.2 网络的聚类系数

假设有向网络中的一个节点 $v_i$ 有 $k_i$ 条边将它和其它节点相连，这 $k_i$ 个节点就称为节点 $v_i$ 的邻居。这 $k_i$ 个节点之间实际存在的边数 $t_i$ 和总的可能边数 $k_i(k_i-1)$ 之比，定义为节点 $v_i$ 的聚类系数：$C_i$，$C_i=\frac{t_i}{k_i(k_i-1)}$；整个网络的聚类系数 $C$ 就是网络中所有节点聚类系数 $C_i$ 的平均值。

随机网络中的任意一个节点，它的两个相邻节点彼此连接的概率不会高于网络中随机选择的任意两个节点的相连概率。因此随机网络的集簇系数是：$C_{rand}\approx\langle k\rangle/N$，其中 $\langle k\rangle$ 是网络的平均度，$N$ 是网络中的总节点数。对于小世界网络，其集簇系数 $C>>C_{rand}$。

聚类系数反映的是相邻节点之间联系的紧密程度。从表 2 可以看到，在 8 个软件依赖网络中 $C>>C_{rand}$，这说明软件依赖网络同样具有较高的聚类系数。这符合软件开发中提倡的高内聚原则。

3.2.3 网络的平均距离

网络的平均距离 $d$ 是网络连接程度的全局指标：$d=\frac{1}{N}\sum_{\forall i,j}d_{ij}$，其中 $d_{ij}$ 是网络中任意两个节点 $v_i$ 和 $v_j$ 之间的最短距离。

对于随机网络，平均距离 $d_{rand}\approx\frac{\log N}{\log[k]}$，小世界网络的平均距离 $d$ 与随机网络的平均距离相似，即 $d\approx d_{rand}$。

对比 8 个软件包的依赖网络可以看到，它们的平均距离约等于同规模随机网络的平均距离，即 $d\approx d_{rand}$。

网络的平均距离是网络整体特性中一个非常重要的方面。对于平均距离较大的网络，信息在网络节点中传播开来需要较长的时间。而对于平均距离较小的网络，网络的连接会相对较强，不容易被破坏，信息的传递也会更迅速、更稳定和更加准确可靠。

软件依赖网络中节点间的平均距离较小，表示计算服务提供者之间的依赖深度较浅。这有利于加快服务提供者之间信息传递的速度，减少服务的响应时间，提高执行效率。

软件依赖网络中存在较高的聚类系数和较小的平均距离(即小世界特性),说明软件在设计实现时存在大量团体,它们遵循高内聚低耦合的原则,团体内部成员之间联系紧密,而团体之间的联系比较松散。连接不同团体的节点作为软件内部交互的核心,很可能就是软件的骨架,即软件体系结构的核心实现。从拓扑结构出发,寻找并分析这样的节点以及节点所在的团体,有助于我们抽取软件系统的体系结构,理解软件的运行机理。

**结束语** 本文分析了国内外现有的软件静态结构的网络建模方法的特点和不足,给出了软件依赖网络模型的定义。将软件静态结构抽象为有向图,直观地建立了不同粒度软件网络模型之间的映射关系。

其次,以面向对象的 Java 软件为例,对 8 种不同的软件依赖网络的结构特性进行了分析,证实了软件依赖网络具有高聚类系数和平均距离很小的小世界特性,入度基本服从幂率,出度基本服从带指数截断的幂率分布。并针对上述特征,探讨了它们的软件工程含义,以及对软件开发和维护潜在的指导意义。

软件依赖网络建模方法能够适用于不同类型的软件,具有较高的分析效率,能够及时响应软件变更,可以为软件的维护以及逆向工程和再工程分析提供依据。

## 参 考 文 献


[1] Valverde S, Cancho R F, Sole R V. Scale-free networks from optimal design[J]. Europhysics Letters (EPL), 2002, 60(4): 512-517

[2] Myers C R. Software systems as complex networks: Structure, function, and evolvability of software collaboration graphs[J]. Physical Review E, 2003, 68(4): 046116

[3] Valverde S, Sole R V. Hierarchical Small Worlds in Software Architecture[R]. SanteFe Institute, 2003

[4] Potanin A, Noble J, Frean M, et al. Scale-free geometry in OO programs[J]. Communications of the ACM, 2005, 48(5): 99-103

[5] 韩明畅,李德毅,刘常昱,等. 软件中的网络化特征及其对软件质量的贡献[J]. 计算机工程与应用, 2006(20)

[6] Hyland-wood D, Carrington D, Kaplan S. Scale-free Nature of Java Software Package, Class and Method Collaboration Graphs [R]. MIND Laboratory, University of Maryland College Park, 2006

[7] La Belle N, Wallingford E. Inter-package Dependency Networks in Open-source Software[Z], 2004

[8] De M A, Lai Y C, Motter A E. Signatures of small-world and scale-free properties in large computer programs[J]. Physical Review E, 2003, 68(1): 017102

[9] Yutao M, Keqing H, Du Dehui, et al. A Complexity Metrics Set for Large-scale Object-Oriented Software Systems[C] // Computer and Information Technology, 2006. CIT'06. The Sixth IEEE International Conference on, 2006: 189

[10] Amaral L A N, Scala A, Barthélémy M, et al. Classes of small-world networks[J]. PNAS, 2000, 97: 11149-11152

[11] Newman M E. Scientific collaboration networks. I. Network construction and fundamental results[J]. Physical Review E, 2001, 64(1): 016131

[12] DependencyFinder 1.2.0. http://depfind.sourceforge.net/, 2007

[13] Pajek. http://vlado.fmf.uni-lj.si/pub/networks/pajek/default.htm, 2007


(上接第 155 页)

$\theta(\theta(1-x_2,1-y_2),1-y_2)\geqslant 1-x_2\geqslant x_1$

$\theta(\theta(1-x_2,1-y_2),1-y_2)\geqslant 1-x_2$,均成立

所以性质(12)成立。

(13) 将式(5)代入并化简得

$\Phi(\Theta(x,y),z)=(T(\theta(x_1,y_1)\wedge\theta(1-x_2,1-y_2),z_1),1-T(\theta(1-x_2,1-y_2),1-z_2))$

$\Theta(x,\Phi(y,z))=(\theta(x_1,T(y_1,z_1))\wedge\theta(1-x_2,T(1-y_2,1-z_2)),1-\theta(1-x_2,T(1-y_2,1-z_2)))$

由 $\theta$ 的性质得

$T(\theta(x_1,y_1)\wedge\theta(1-x_2,1-y_2),z_1)\leqslant T(\theta(x_1,y_1),z_1)\leqslant\theta(x_1,T(y_1,z_1))$

$T(\theta(x_1,y_1)\wedge\theta(1-x_2,1-y_2),z_1)\leqslant T(\theta(1-x_2,1-y_2),z_1)\leqslant T(\theta(1-x_2,1-y_2),1-z_2)\leqslant\theta(1-x_2,T(1-y_2,1-z_2))$

$\therefore T(\theta(x_1,y_1)\wedge\theta(1-x_2,1-y_2),z_1)\leqslant T(\theta(x_1,y_1),z_1)\wedge T(\theta(1-x_2,1-y_2),1-z_2)$

$T(\theta(1-x_2,1-y_2),1-z_2)\leqslant\theta(1-x_2,T(1-y_2,1-z_2))$

所以性质(13)成立。

(14) 证明与性质(7)类似。

(15) 由性质(3)得

$\Theta(x,y)\leqslant_L{}^*\ \Theta(x,y\vee z),\Theta(x,z)\leqslant_L{}^*\ \Theta(x,y\vee z)$

$\therefore\Theta(x,y)\vee\Theta(x,z)\leqslant_L{}^*\ \Theta(x,y\vee z)$

所以性质(15)成立。

(16) 证明与性质(13)类似。

**结束语** 本文研究了直觉模糊三角模的剩余蕴涵与三角模的剩余蕴涵的关系,从而可以利用三角模的剩余蕴涵来计算直觉模糊三角模的剩余蕴涵;研究了直觉模糊三角模的剩余蕴涵的系列性质,并给出了详细的证明过程,从而进一步夯实了直觉模糊逻辑算子的理论基础,深化了直觉模糊集理论在知识处理领域中的应用研究。

## 参 考 文 献


[1] Deschrijver G, Cornelis C. On the representation of intuitionistic fuzzy t-norms and t-conorms [J]. IEEE Trans. Fuzzy Systems, 2004, 12 (1): 45-61

[2] Deschrijver G, Kerre E E. Implicators Based on Binary Aggregation Operators in Interval-valued Fuzzy Set Theory[J]. Fuzzy Sets and Systems, 2005(153): 229-248

[3] Pankowska A, Wygralak M. General IF-sets with triangular norms and their applications to group decision making [J]. Information Sciences, 2006 (176): 2713-2754

[4] De S K, Biswas R, Roy A R. Some operations on intuitionistic fuzzy sets [J]. Fuzzy Sets and Systems, 2000(114): 477-484

[5] 张文修,吴伟志,梁吉业,等. 粗糙集理论与方法[M]. 北京:科学出版社, 2006: 168-170

[6] 雷英杰,王宝树. 直觉模糊关系及其合成运算[J]. 系统工程理论与实践, 2005, 25(2): 113-118

[7] 雷英杰,王宝树. 基于直觉模糊逻辑的近似推理方法[J]. 控制与决策, 2006, 21(3): 305-310

[8] 雷英杰,王宝树. 直觉模糊集时态逻辑算子与扩展运算性质. 计算机科学, 2005, 32(2): 180-181, 225

[9] 李晓萍. 关于三角模的直觉模糊群及其同态像[J]. 模糊系统与数学, 2005, 19(1): 57-62