一个支持软件重用的信息检索系统—KDZ

张少平 王怀民 陈火旺

(国防科技大学计算机科学系)

AN INFORMATION RETRIEVAL SYSTEM FOR SUPPORTING SOFTWARE REUSE—KDZ

Zhang Shaoping, Wang Huaimin and Chen Huowang

(*Department of Computer Science, Changsha Institute of Technology*)

ABSTRACT

Efficiently organizing and retrieving software component is the key to software reuse technique. The paper introduces an information retrieval system for supporting software reuse—KDZ. The system bases on the formal description of software component, and takes the reusability theory as the rule of verification. By the Combination of term-rewriting and retrieving technique, the KDZ system has the function of semantics verification.

摘 要

软构件的有效组织与检索是软件重用技术的关键。本文介绍了一个支持软件重用的信息检索系统—KDZ。该系统以形式化方法描述软构件为基础，以重用理论为验证准则，通过项重写技术与检索技术相结合，使KDZ 系统具有语义验证之功能。

§1. 引言

在软件工程领域中，人们迫切地希望找到一种软件开发方法，提高软件生产效率，减少软件开发代价，且使软件的可靠性得以保证，从而缓解软件危机。软件重用似乎是

1990 年1 月7 日收到，1990 年4 月20 日定稿。

一种行之有效的方法，实践表明，由初始规范完全自动地产生满足规范的程序变换方法是非常困难的。

尽管软件重用思想早已为人们所接受，然而软件重用技术并没有在实践中得到广泛的应用，特别是对于开发大型、复杂的软件系统。为什么会出现这种情况呢？这里面除了一些人为因素外，还存在一些技术方面的问题，如软构件的有效组织与检索、软构件的合成技术等等。软件重用技术的广泛应用，有待于这些技术问题的解决。

为了解决这些技术问题，我们在软构件的有效组织与检索方面进行了尝试，建立了一个支持软件重用的信息检索系统—KDZ，并且在Sun 工作站上，我们已经着手开发了该系统。本文主要介绍KDZ 系统的设计思想。在 §2 中，我们讨论了软构件的组成；在 §3 中提出了软件重用理论；§4 讨论了软构件的组织与检索；§5 讨论了系统中项重写子系统对软构件检索的支持，而在最后，我们讨论了结论以及今后的工作。

§2. 软构件的组成

软构件的定义依赖于不同的重用系统，它既可以是完成某一功能的程序代码，也可以是软件工具等。在KDZ 系统中，我们定义软构件为完成某一功能的程序代码，因为KDZ 系统是用来依据软件重用技术支持大型软件开发的。

一般而言，设计一个可重用软构件要比设计一个非重用的程序代码要难。因为我们在设计可重用的软构件时，需要考虑许多因素，比如应把一些功能相关的程序段组装成一个软构件，保证软构件与环境的相关性达到最小，使软构件具有代表性及标准化等等。

只有在清楚了软构件的功能时，我们才能重用软构件，这就意味着我们应以某种方式描述软构件。因为对于一段程序代码，我们很难理解其语义，显然，程序代码本身不能用来作为组织和检索软构件的对象。

在当前软构件的检索系统中，比较流行的是用关键词或自然语言描述软构件。通过关键词来描述软构件的方法易于实现，但它的缺点也十分明显。关键词对软构件的功能描述不够精确，并且我们也很难对检索出的信息数量进行控制。虽然用自然语言能够比较容易地对软构件的语义进行描述，但它的语义描述不够精确，具有二义性，且该方法不易于实现。在KDZ 系统中，我们用形式规范对软构件进行描述，即EX-ML 形式的代数规范(有关EX-ML 语言的详细文法，请参见[8])。规范由语法、语义两部分组成，语义采用等式公理形式。我们可以把软构件抽象地表示成下述二元组：Component=<sp, code>，其中sp 为EX-ML 规范，code 为程序代码。程序代码一般用Ada 语言编写，并且假定通过反复验证和调试，程序代码code 满足规范sp。对于规范sp，我们还可以进一步地划分，把其划分成语法说明和公理描述两部分，即sp=< Σ , E>，其中Σ 为标记集合，E 为算式的集合。

例如，假设我们要设计一个实现整数表的并置和倒置的软构件，对于该软构件可进行如下规范描述：

```
sp=
    sig
    Σ { type intlist = nil |int::intlist
        val append : intlist * intlist → intlist
                rev : intlist → intlist
        end
    E { axiom
            append (nil, l)=l
            append (x::l, m)=x::append(l, m)
            rev (nil)=nil
            rev (x::l)=append (rev(l), x::nil)
        end
```

§3. 重用理论

在重用软构件时，我们必须遵循一定的准则，而不能毫无标准地随意重用，这就意味着用户对软构件的需求描述应与被重用的软构件之间存在某种语义相似性，即存在重用理论作为重用标准。

重用一般可分为“黑箱重用”和“白箱重用”两种。“黑箱重用”是指不需要对软构件做任何改动，直接应用于新的上下文中，而“白箱重用”则是指软构件的实体必须经过修改后，才能应用于新的重用环境中。“白箱重用”可能要涉及到软构件的语义修改。在KDZ 系统中，我们仅考虑只进行词法变换，而不涉及语义更改的简单情况。

现在的问题是sp_0 为用户需求规范，sp 为已实现了的软构件规范描述，那么该软构件是否可作为实现规范sp_0 的软构件呢? 首先我们定义标记等价:

定义1 设Σ_1、Σ_2 为两个代数规范的标记，R_Σ 为标准词汇的项重写系统，我们称标记Σ_1、Σ_2 在R_Σ 上等价，$\Sigma_1 \underset{R_\Sigma}{\equiv} \Sigma_2$ 是指Σ_1、Σ_2 中的所有项，在R_Σ 的作用下，汇合于同一标记Σ, 即$\Sigma_1 \underset{R_\Sigma}{\longrightarrow} \Sigma$, 且$\Sigma_2 \underset{R_\Sigma}{\longrightarrow} \Sigma$。

例如: $\Sigma_1 = (\{s_0, s_1\}, \{f : s_0 \to s_1\}), \Sigma_2 = (\{s_0, s_2\}, \{g : s_0 \to s_2\}), R_\Sigma = \{s_1 \to s, s_2 \to s, f \to +, g \to t\}$。

易证: $\Sigma_1 \underset{R_\Sigma}{\longrightarrow} \Sigma, \Sigma_2 \underset{R_\Sigma}{\longrightarrow} \Sigma$, 其中$\Sigma = (\{s_0, s\}, \{t : s_0 \to s\}$。

从而我们可以得到$\Sigma_1 \underset{R_\Sigma}{\equiv} \Sigma_2$。

当标准词汇的项重写系统(也称作语法项重写系统) 作用于公理时，也将公理中的非标准词汇替换成标准词汇。

若$t = s \in E$, R_Σ 为语法项重写系统，则$R_\Sigma(t) = R_\Sigma(s) \in E'$ 表示重写后的等式公理。

有了上面的定义准备，现在我们可以定义重用理论。

定义2 设sp_0 为用户提出的(不完全的或部分的) 需求规范，R_Σ 为标准词汇的项重写系统，我们说软构件<sp, code> 中的规范sp 与sp_0 相似是指:

(a) $\Sigma_0 \underset{R_\Sigma}{\equiv} \Sigma$

(b) $\forall t = s \in E_{sp_0} \Longrightarrow I_{sp} \models R_\Sigma(t) = R_\Sigma(s)$, 其中$I_{sp}$ 为由规范sp 所确定的初始模型。

如果规范sp 与sp_0 相似，我们就可以重用程序代码code 作为实现规范sp_0 的程序代码。

例如，用户可能写一个关于add, rev 的不完全规范sp_0:

```
sp0=
    sig
      type charlist=nil | char*charlist
      val add: charlist*charlist→charlist
           rev: charlist→charlist
      end
      axiom
           add (x::l, m)=x::add (l, m)
           rev (rev(l))=l
    end
```

可以验证sp_0 与sp 相似(验证方法请参见 §5 语义项重写子系统的讨论)，这时我们就认为用户需求规范sp_0 的意图在于寻找表的并置与倒置操作的软构件。

§4. 软构件的组织与检索

为了能够有效地进行软构件的检索，必须对软构件分类与组织，把所有的软构件组成一个可重用的软构件库。该构件库必须是一个开放型的，易于扩充和修改，并且当需要时，能够对软构件库中的软构件进行浏览、选择及检索等操作。

为了达到上述目标，我们需要建立一个结构化的重用软构件库。每一个软构件都具有分类特征，这些特征即是软构件中的类型及操作描述。图1 说明了一种“类型特征”和“操作特征”。

图1　软构件的类型特征和操作特征

依据类型特征和操作特征即可在软构件集合上建立一个格结构，图2 说明了基于类型特征和操作特征所建立的一般化格结构。

软构件的格结构使我们能够有效地浏览重用软件库中的软构件，且通过把软构件集合划分成有意义子集的方法，使我们能够有效地进行软构件检索。例如，如果需要包含表连接的软构件，则属于格中list-append 节点的软构件子集为我们所感兴趣的；如果我们只考虑包含并置操作的软构件，而不管操作类型，则我们只检查append 节点的软构件集合。

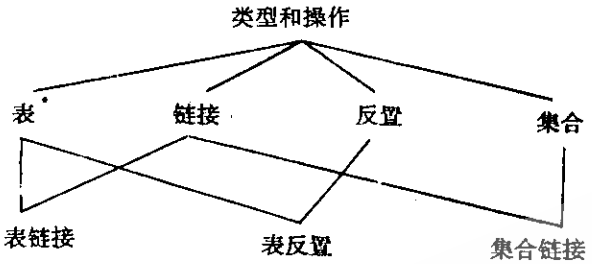

图2 一般化的格结构

在建立软构件库时，为了节省存贮空间，提高软构件的检索效率，库中的软构件的规范描述都用标准词汇书写，例如change、modify 及updata 都表示相似的意义，在建立软构件库时，我们把updata 定义成标准词汇，若规范中需要这些词汇时，则只用updata 来表示。当用户进行软构件检索时，可不受标准词汇的限制。语法项重写子系统自动进行标准词汇变换。

同一个程序代码，可以由许多语法形式不同、但语义相同的代数规范描述。同样，我们对于重用软构件库中软构件，也定义一个标准形式的规范描述。标准形式的规范应满足Huet-Hollet 的项重写系统的完全条件。语义项重写子系统根据重用理论进行语义检查，即验证软构件是否可为用户所重用。

一般而言，软构件的检索过程可分二步:

(1) 对重用软构件库进行访问。依据用户的需求，提取一个或几个软构件。

(2) 验证所提取的软构件是否满足用户的语义要求，即证明用户需求规范是否与软构件的描述规范相似。

在对软构件库进行访问前，应对用户的需求规范进行词汇标准化操作，即把需求规范转换成标准形式的规范，这一过程是通过语法项重写子系统实现的。在对软构件库进行访问，提取软构件时，KDZ 系统把用户需求规范的语法描述作为一个需求模式，在软构件库中进行模式匹配。只有当需求模式与软构件的语法模式完全匹配时，该软构件才被提取。因为依据重用理论，相似的规范应满足语法的一致性。作为软构件检索过程的第二步，语义验证是以需求规范的公理为基础，通过语义项重写子系统措施的。

§5. 项重写子系统

在KDZ 系统中，项重写子系统有二个，一个为语法项重写子系统，而另一个为语义项重写子系统，它们分别作为标准形式的规范转换及语义验证的辅助工具。

在建立重用软构件库时，我们也建立了一个标准词汇与非标准词汇的对应表，例如: