SAIS: 一个基于规则的软件结构分析集成系统

费翔林 王和珍 汪承藻 魏红 朱根江

(南京大学计算机软件研究所, 210008)

SAIS: AN INTEGRATED RULE-BASED SYSTEM FOR ANALYSIS OF SOFTWARE STRUCTURE

Fei Xianglin, Wang Hezhen, Wang Chengzao, Wei hong and Zhu Genjiang

(Institute of Computer Software, Nanjing University, Nanjing 210008)

ABSTRACT

As a field of research in software engineering, software Re-Engineering has started to attract wide attention in recent years.The key to the Re-Engineering activity is the ability to recover "lost" or otherwise unavailable information from the information available in the existing software source code.This paper introduces SAIS, An Integrated Rule-based System for Analysis of Software Sructure, which is designed to provide partially support to software maintenance and reverse-engineering activities. The design, implementation, application and features of SAIS are briefly described in the paper.

摘 要

作为软件工程研究的领域之一，软件再工程 (Re-Engineering) 在最近几年引起了人们广泛的关注. 软件再工程的关键是：从现有原代码所提供的信息中去获得和恢复丢失的系统设计和规格说明信息的能力. 本文介绍了 SAIS, 一个基于规则的软件结构分析集成系统，它被设计用于提供对软件维护和软件逆向工程活动的部分支撑. 文章简要描述了 SAIS 的设计、实现、应用和特点.

本课题获得国家“七五”重点科技攻关项目的资助. 作者 **费翔林**，南京大学副教授，系副主任，主要研究领域为软件工程，包括面向对象计算和 Re-3 技术. **王和珍**，南京大学副教授，主要从事 Re-3 和情报检索方面的研究工作. **汪承藻**，南京大学高级工程师，主要研究领域为 Re-3 技术和操作系统. **魏红**，1983 年在南京大学获硕士学位，现任讲师，主要从事 Re-3 技术和数据库方面的研究工作. **朱根江**，1991 年在南京大学获硕士学位，主要从事 Re-3 技术方面的研究工作.

§1. 引 言

Re-3 技术 (Re-Structuring 、 Reverse-Engineering 、 Re-Engineering) 是近年来发展起来的计算机技术新领域. 迄今为止, 全世界积累了大量软件, 已经使用和正在开发的软件约有 1000 亿行源码. 它们多数是使用过时的技术生产的, 存在着非结构化、含有无效代码和冗余码、打过补钉以及文件说明和源码不一致等缺陷. 据估计, 目前多数国家中, 软件人员的 60%—80% 的时间花费在软件维护上 [1][2][3]. 所以, 对现有的软件就存在着发掘整理和推陈出新的问题. Re-3 就是用于革新和改造现有软件的新技术, 它向软件注入新的活力、校正其错误、扩充其功能、改良其结构、改进其性能, 以此延长软件的寿命, 发挥更大效益, 创造更多价值.

Re-3 技术, 是指:

· 再结构 (Re-Structuring)

将非结构化的程序或经过多年修补结构不良的程序, 通过软件工程方法和工具重新整理和改造, 使之具有清晰合理的结构, 或转换成结构化程序. 已有许多方法可以完成软件再结构任务.

· 再工程 (Re-Engineering), 它又分:

· · 逆向工程 (Reverse-Engineering)

对现有软件进行分析理解, 弄清工作原理, 恢复设计方案, 抽象出其结构和功能, 采用高层次方式精确地描述该软件. 由于程序不断地被修补, 真正能确切描述程序当前功能的应该是源代码本身, 所以, 逆向工程中, 分析理解的对象通常是源代码.

· · 正向工程 (Forword-Engineering)

在逆向工程的基础上, 通过已经恢复、抽象和精化的高层描述, 采用现代软件工程方法和工具, 重新开发出与原有软件功能等价, 结构良好、效率较高、易于维护的软件.

软件再工程活动的关键是: 从现有源代码所提供的信息中去获得和恢复被丢失的系统设计与规格说明信息的能力, 这是软件逆向工程的主要任务. 然后, 在软件正向工程中完善并扩充现有系统的功能. SAIS(An Integrated Rule-based System for Analysis of Software Structure) 属于逆向工程范围, 它是帮助程序员分析理解大型程序的辅助工具系统, 主要用于:

· 对现有大量软件的发掘整理、推陈出新, 做到“古为今用”, 充分利用软件资源;

· 对进口软件进行理解、消化、改造和扩充, 做到“洋为中用”, 这是一条投资少、见效快的软件开发途径;

· 作为辅助教学工具, 帮助学生学习软件课程.

§2. SAIS 的设计

2.1 系统概述

SAIS 是在 SUN3 系列工作站上实现的面向多语言的软件结构分析集成系统. SAIS 处理的对象是大型、复杂或不熟悉的源程序, 目前面向 PASCAL 和 FORTRAN 语言书写的程序. SAIS 提供分析、观察、追踪、回溯、统计、文档生成等一系列功能和设施,

帮助软件人员模拟人工分析，理解大型软件的全过程，达到向软件人员提供软件全貌、观察程序细节并能建立各种文档供使用．此外，还为定量研究程序可靠性、复杂性提供静态特性和统计数据． SAIS 的使用减轻了软件人员分析和理解软件的智力负担，提高了软件理解的生产率．

2.2 设计原则

SAIS 遵循以下设计原则：首先，系统应面向多语言．因而，不同语言书写的靶程序知识的一致性内部表示必须考虑．此外，为了支持软件分析理解的全过程，形成一个功能齐全，有统一用户界面，便于新手程序员使用，适合不同应用的集成系统，还着重考虑了：设施多样化、接口友善化、工具集成化、系统开放化和结构模块化等因素．

2.3 功能设施

SAIS 系统具有如下功能与设施：

·程序结构分析

设计一个程序的重要方面是设计它的结构，理解一个程序的首要任务就是理解它的结构．程序结构涉及程序单位及由程序单位构成一个程序的方式， SAIS 可以从总体上弄清一个程序由哪些程序单位构成，各部分之间如何相互联系，并以程序结构树简明地刻划其概貌．

·控制流跟踪

控制流分析能对控制传递和流向进行跟踪或回溯，查明控制从何处来，又转向何处去，受控于哪些条件等，此过程可逐步细化，由外部关系到内部细节，从中获得对程序更深的理解．控制跟踪的功能和设施包括：过程调用，过程被调用，源和目标过程之间语法路径搜索，隐蔽和不隐蔽内嵌语句的程序控制流程图等观察．

·数据流分析

分析程序中所有变量的定值和引用之间的关系称为数据流分析．控制流分析掌握程序中控制的传递和流向，而数据流分析则在控制流分析基础上分析程序中各个变量的演变过程，如变量在何处定值，在何处被修改，又在何处被使用等．数据流跟踪包括：过程数据结构，过程数据接口，公共数据，隐蔽或不隐蔽内嵌语句的程序源码，数据使用—定值点等观察．

·复杂性度量

程序理解的困难程度与程序的复杂性密切相关，程序复杂性度量就显得十分重要，这种度量可归结为：结构复杂性、运算复杂性、数据复杂性和执行程序长度． SAIS 在分析中可统计变量的类型、个数，运算符的类型、个数，以及定值—使用链的个数；在控制流分析中可以给出结点、路径、循环、嵌套等特性．此外，还给出程序行数、调用深度、数据类型等与程序复杂性有关的信息．

·文档保存

在分析理解过程中，可以以交互方式在屏幕上显示结果，也能通过硬拷贝输出所需的正文和图形文档资料，供以后使用 [4][5][6]．

§ 3. SAIS 系统的实现

3.1 系统模块和组成

SAIS 系统由三大模块组成: 多语言接口、基于规则系统和系统集成模块.

(1) 多语言接口模块

有两部分: 词法分析程序生成器, 用于生成特定对象语言词法分析程序; 对象语言的语法/ 语义规则库和靶程序知识获取程序, 实现基于规则的语法分析及句子识别, 自动抽取靶程序知识, 组织并存储到事实库.

(2) 基于规则系统模块

是 SAIS 系统的核心, 它包括:

· 事实库　分为后援事实库和动态事实库, 前者存储所有从靶程序中获取的程序知识 — 事实; 后者存放从后援事实库中调入的当前使用的事实.

· 规则库　存放有关获取靶程序知识的知识及分析理解软件的一般性 (经验) 知识.

· 推理机　就是 Prolog 编译器本身, 推理过程是目标驱动的, 由于其优良的说明性风格和极强的模式匹配能力, 适宜于构造 SAIS 系统.

· 控制接口　实现提问分析、请求和接收用户输入数据、激发规则执行、启动推理机工作和控制结果输出.

(3) 系统集成模块　是 SAIS 的总控模块, 提供多窗口、菜单、编辑、编译、图形及各种系统功能支撑; 实现用户接口管理; 它还将系统各部分组成一个完整的集成系统.

SAIS 系统的组织如图 1 所示.

3.2 知识表示与获取

设计一个基于知识的系统, 应该根据: 应用领域、系统环境、表达能力、实现效率等多种因素折衷考虑以选择一种易于产生、修改、理解、使用的知识表示. 在 SAIS 系统中, 事实用谓词来表示, 每个谓词可以包含一个或多个对象. 事实的一般形式为:

谓词名 (对象 1, 对象 2, ···, 对象 n)

例如, 谓词

Call (Statement-no,Procedure1,Procedure2)

描述过程调用关系. 其中, 三个对象分别表示调用关系的产生点, 调用过程号和被调用过程号.

规则用于描述事实之间的关系, 一般形式为:

IF < 前提 > THEN < 结论 >

其中, 前提由条件组成, 条件可以是一个或多个, 多个条件时用 and 组合条件; 条件可以是外部过程或操作命令, 也可以是谓词; 谓词可以是内部谓词 (由 Prolog 语言提供). 也可以是外部谓词 (由用户定义). 仅当某规则被激活, 前提条件全部成立时, 才能获得相应的结论. 结论是从靶程序中抽出或由一组事实导出的新事实. 通常, 这一事实是对事实库执行一组动作的结果. 使用 BNF 范式, 规则可表示为:

< 规则 >	::= (IF < 前提 > THEN < 结论 >)
< 前提 >	::= (AND {< 条件 >})
< 条件 >	::= (< 谓词 >|< 操作命令 >|< 外部过程 >)
< 操作命令 >	::= <Unix 系统调用 >
< 外部过程 >	::= <C 语言过程 >
< 结论 >	::= { < 谓词 >}

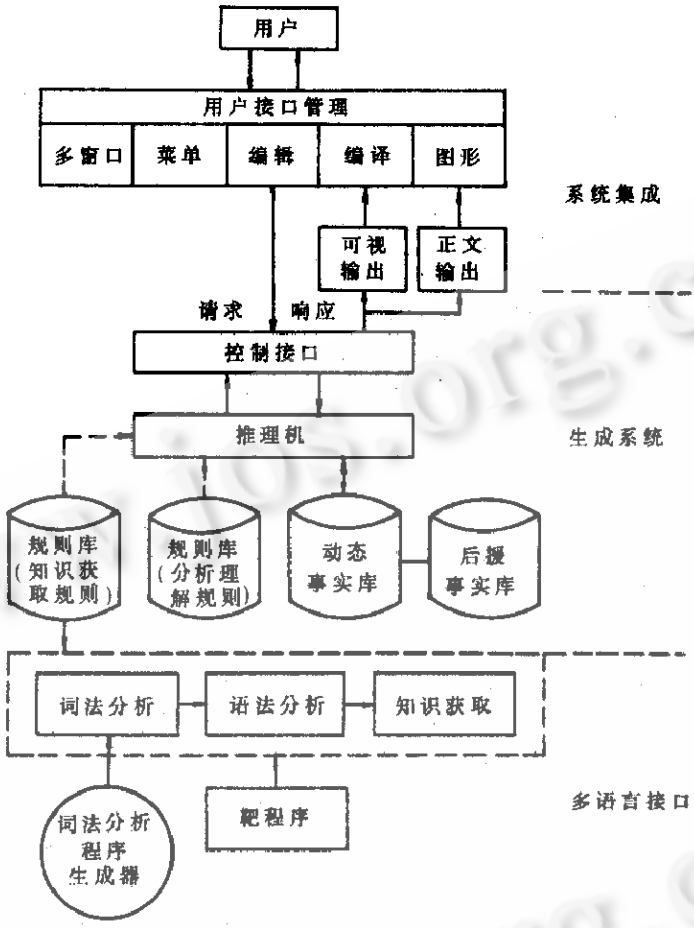

图 1　SAIS 的组织

< 谓词 >　　::= (< 内部谓词 > | < 外部谓词 >)

当规则的前提条件成立时，则此规则可用，规则的激活往往会增加动态事实库的内容.

例如，下面两条规则用于自动获得基类型

如果 (1) 是用户定义的变量，并且

(2) 是类型变量，并且

(3) 基类型

则　获得该基类型.

如果 (1) 是用户定义的变量，并且

(2) 是类型变量，并且

(3) 非基类型

则　查该非基类型的基类型

各种知识获取辅助手段已被开发，仅有极少数系统能实现自动化知识获取. 很幸运，因 SAIS 面临应用问题的特殊性，使依靠计算机全自动地获取靶程序知识成为可能. 由

于要求输入给 SAIS 的靶程序在语法上是正确的，从而，使靶程序的知识获取工作较为简单．每当获取到新的知识，便进行约束性检查，凡符合约束条件的事实方可加入事实库．例如，对于 FORTRAN 语言，判别隐含整型类型说明的约束性规则为：

如果 (1) 变量为隐含说明

(2) 变量标识符的头字符属于集合 {I-N} 中的一个

则　该变量的类型为整型

SAIS 使用 Unix 操作系统下的 C 语言，Quintus Prolog 语言以及 PS 语言混合编写．C 语言实现词法分析程序生成器、控制接口、系统集成；Prolog 实现知识获取和基于规则分析理解；PS 语言实现激光打印机图形输出 [7][8][9].

§ 4. SAIS 的应用

使用 SAIS 分析理解程序大致执行下列步骤：

·选择程序　获取知识

交互型输入靶文件名，系统自动获取程序知识并存入知识库待用．

·识别结构　实现分治

SAIS 可帮助软件人员弄清大型、复杂程序的全貌．首先，可用层次结构树命令刻划程序总框架，诸如程序包含多少过程和函数，各自在哪层被说明，最大深度多少，各叫什么名字，等等．然后，可用其它结构分析命令继续识别程序结构：一个过程去调用哪些过程，又被哪些过程调用，一个过程到另一个过程的语法路径及调用条件，每个过程的局部数据结构和环境数据接口等等．一个结构化程序应该分而治之，利用 SAIS 可实现对大型程序的分治，让软件人员在相对较小的程序单位上逐个分析理解．

·分析观察　掌握细节

接着，可深入观察程序的过程特性．使用 SAIS 提供的观察、追踪、回溯、抽象、细化等设施作交互型分析理解．可指定过程名、语句或行号定位到欲分析理解的过程．根据控制流程图，可找前一个语句，后一个语句；找前一个过程，后一个过程；可指定隐蔽或细化内嵌语句，获得不同级别抽象的流程图．数据流跟踪能在源代码一级，按逻辑和物理次序找到指定变量的前一个定义点、引用点；后一个定义点、引用点；也可列出程序或过程变量的所有定义点或引用点；可指定隐蔽和细化内嵌语句，获得不同抽象级别的源代码．利用 SAIS 提供的多窗口和多重观察命令，可方便地集中程序的相关特性和不同侧面，如流程图和源码或过程调用和源码的多重观察，更有效地帮助理解程序过程特性和掌握过程细节．

·归纳综合　理解程序

反复使用自顶向下和自底向上的系统分析法，由粗略到具体，观察和分析程序的逐步求精历史；由具体到抽象，在对程序实现细节了解的基础上，运用归纳和综合，对程序进行抽象．通过上述步骤，程序员对程序结构有了较好的理解，此外，SAIS 中，层次结构的逻辑注解作为基本的功能抽象设施，可把对程序的理解综合成注解，系统提供交互型观察，填充和修改注解的功能．当对整个程序作了比较满意的逻辑注解时，就认为达到了对程序的理解．

· 文档资料 拷贝保存

交互型分析观察的同时，可硬拷贝文档资料备用.

已经使用 SAIS 分析了若干大型 PASCAL 和 FORTRAN 源程序，其中，最大一个实例为 PASCAL 可移植编译程序 P4[10]，格式化后程序为 6588 行；程序过程数为 101 个；最大过程说明深度 6 级；主程序实际调用过程数 9 个. 有关其它分析理解的结果见参考文献 [11].

§ 5. 有关工作和 SAIS 的特点

MAP 是美国 Amdahl 公司开发的 COBOL 源程序理解工具，具有分析、追踪、控制和拷贝等设施，帮助完成理解程序的六项任务：识别程序结构，追踪控制流和数据流，了解数据别名，寻找正文和版本控制等. RXVP80 是 VAX 机上配置的 FORTRAN 源程序理解工具，能作静态检查、分析和统计. 提供程序的模块结构和数据流，便于程序修改，指示修改程序可能出现的副作用；以及具有查出未初始化变量或只定义未使用变量等功能. OMEGA 是分析 C 语言数据流的模型系统，能检测公用编程错误的某些类型或确认它们的存在. 类似的系统还有：DAVE, FACES, FAST, TAUS 等 [12].

首先，从应用功能上看，上述工具均面向单语言，不能同时处理多种语言编写的源程序，因而难度较低、易于实现. 其次，从实现技术上来看，在程序分析过程中需要提取、处理和保存大量程序信息 (知识), 采用什么样的数据形式来记录分析过程中获得的各种程序信息是程序分析系统中应解决的一个主要难题. 上述各工具均沿用传统的软件工程技术，采用表格形式记录各种程序信息，这种方式的缺点是零乱、分散、难以反映各种程序信息之间的有机联系，表格的维护工作量大. 当要求处理多语种源程序时，这些缺点就更为突出. 此外，从界面上看，上述工具没有将各种功能和设施集成为一个系统，并充分利用流行的各种接口技术，总的说来，人机界面较差.

与上述工具相比，首先，SAIS 面向多语言，目前已实现了对两种高级语言源程序的分析理解，实现面向多语言的通用分析理解工具较为困难，通常，不能直接从源程序出发，而应从中间代码理解着手，将用高级语言编写的源程序通过词法/ 语法分析转化成通用的中间代码，最后，在中间代码的基础上完成对程序的结构或功能分析. 因而，寻找合适的通用中间代码就成为开发面向多语言程序分析理解工具的关键之一. 在 SAIS 中，提出了一种程序知识表达法 — 谓词表达法，作为不同语种的源程序程序知识的中间代码以实现一致性内部表达，一改过去国内外有关程序分析理解工具都有一定语言限制的状况. 它将人工智能的知识表示和软件工程技术有机地结合起来，为软件工具中长入智能成分做了有益的探索. 程序谓词表达法的主要优点是：逻辑结构清晰、表达方式统一、维护管理简单、扩充语种方便，更重要的是程序知识的表达与编程语言无关，建立程序信息之间的联系方便，容易将基于规则技术应用于源程序的句子识别和语法分析.

其次，由于 SAIS 面向的应用问题的特殊性，加上采用合适的知识表示，从而能全自动地实现基于规则的源程序知识获取，提高了系统的运行效率. 在程序知识采用谓词表达的基础上，开发和建立了一个具有一种软件理解方法学、一组应用工具和一个知识库支撑的软件结构分析集成系统.

最后，从界面上看， SAIS 不仅将应用系统的各种功能，而且还将编辑、编译、运行、操作系统命令调用等各种系统功能和工具融为一体，组合成集成系统，提供了统一的内部接口，菜单引导、鼠标驱动、多窗口支撑、交互型工作、信息提示等设施提供了极为友善的用户界面，以达到方便用户使用的目的.

综上所述， SAIS 的主要特点可以归结为：集成系统、界面友善；功能多样、自给自足；面向多语言、提供多设施；系统开放式、功能可增减；谓词表达法、语种易扩充；知识自动取、系统效率高.

§ 6. 结束语

Re-3 技术已引起人们特别关注，现有的诸如控制流分析、数据流抽取、复杂性测试和程序调整等商品化工具，尽管十分有用，但因不涉及该软件所属的领域知识，故不可能从高层次全面地分析理解它. 发展方向应使系统能描述和解释软件所涉及的问题领域. 再工程的工具或环境，将来不仅要具有软件工程方面的知识，而且更重要的是应获取软件描述的问题所涉及的领域知识，换句话说，未来应该建立和开发的是各种各样面向专业领域的 Re-3 工具和环境.

致谢：在 SAIS 的研制过程中对徐家福教授的热诚指导，特表示诚挚的谢意. 还要衷心感谢参加本项目设计、编程和调试工作的顾卫明、李保东、洪朝晖、陆远等同志.

参考文献

[1] G.M.E.lafue (organizer), Pannel: Software Re-Engineering, 12th International Conference on Software Engineering, March 26-30, Nice, 1990.

[2] Greham Mead, Re-3: Re-Structuring, Reverse-Engineering and Re-Engineering, The Means to Renovate Old System, Hong Kong Computer Conference 1990, May 5-7, Hong Kong, 1990.

[3] Wojtek Kozaczynski and Jim Q.Ning, SRE: A Knowledge-Based Environment for Lardge-Scale Software Re-Engineering Activities, 11th International Conference on Software Engineering, 1989.

[4] Fei Xianglin etc., TAUS: A File-Based Software Understanding Tools, Journal of Computer Science and Technology Vol. 5 No. 1, 1982.

[5] Steven S.Muchnick and Neil D.Jones, Program Flow Analysis: Theory and Applications, Prentics-Hall, Inc., Englewood Cliffs, New Jersey, 1981.

[6] 费翔林、王和珍、汪承藻，基于文件和基于知识的软件理解，计算机研究与发展， Vol.26, 1989.

[7] F.Hages-Roth etc., Building Expert System, Addisson-Wesley Publishing Company, Inc., 1983.

[8] Sun View Programmer's Guide, Sun Microsystems,Inc., 1987.

[9] Quintus Prolog Library Manual, Quintus Computer System, Inc., 1987.

[10] 杨德元， Pascal 可移植编译程序 P4 及其分析说明，清华大学出版社， 1982.

[11] 基于规则的软件结构分析集成系统《技术报告》，南京大学计算机科学系， 1989 年 6 月.

[12] 朱三元、刘胄，国内外软件逆向工程综述，计算机应用和软件， vol.7, No.4, 1989, 1-11.