

算法演示引擎的设计与实现

苏莹, 吴伟民, 黎凯伦

(广东工业大学 计算机学院, 广东 广州 510090)

摘要: 算法演示软件通过图形的方式直观形象地展示算法的执行过程, 为理解算法、学习编程和调试程序提供了便利。算法演示引擎(AAE, Algorithm Animation Engine)为算法演示软件的开发提供了一种解决方案。通过构造算法演示引擎, 可以实现交互式的算法动态演示。该算法演示引擎已在本工作室开发的算法演示软件中应用, 取得良好的效果。

关键词: 算法演示引擎; 算法动态演示; 算法演示软件

中图分类号: TP311.1

文献标识码: A

文章编号: 1673-629X(2006)07-0049-03

Design and Implementation of Algorithm Animation Engine

SU Ying, WU Wei-min, LI Kai-lun

(Computer Faculty, Guangdong University of Technology, Guangzhou 510090, China)

Abstract: Algorithm animation software displays the dynamics of algorithms specified in textual form through their graphical presentation, helping users understand algorithm, learn and debug programming. Algorithm animation engine is one way to develop algorithm animation software. Interactive algorithm animation can be implemented by constructing algorithm animation engine. This algorithm animation engine has been used in one algorithm animation software developed and has a good result.

Key words: AAE; algorithm animation; algorithm demo software

0 引言

数据结构是计算机软件理论和技术的重要基础, 不仅是计算机各专业的核心课程, 而且是其他理工专业的热门选修课。该课程主要讨论抽象数据关系和算法的定义、表示和实现, 用常规的板书或一般的幻灯投影授课均难以有效地展示数据结构和算法的抽象性和动态性, 容易造成教学低效和学时膨胀^[1]。

算法演示软件通过图形的方式直观形象地展示算法的执行过程, 为理解算法、学习编程和调试程序提供了便利。到目前为止, 国内已经有许多在各种平台上开发的算法演示软件^[2-5]应用在教学方面, 并且取得了显著效果。实现算法演示软件的关键是如何实现动态演示。算法演示引擎 AAE (Algorithm Animation Engine) 为算法动态演示的实现提供了一种解决方案。通过构造 AAE, 可以实现交互式的算法动态演示。应用 AAE 开发算法演示软件, 提高了软件开发的效率。

1 AAE 的总体设计

算法演示引擎 AAE 是这样—个程序: 系统运行时将—组算法的代码交给 AAE 处理, AAE 将每一行代码与数

据结构演示 API 中的画图操作进行匹配, 然后执行该行代码所触发的画图操作。

因此, 一个基于算法演示引擎 AAE 的应用包括以下 3 个部分(图 1 是一个应用例图)。

- * AAE: 它负责将描述算法的代码与演示图形联系起来, 然后执行相应的画图操作;
- * 代码库: 保存描述算法的所有代码;
- * 数据结构可视化类库: 提供数据结构演示 API, 保存所有对应每一行代码所触发的画图操作。

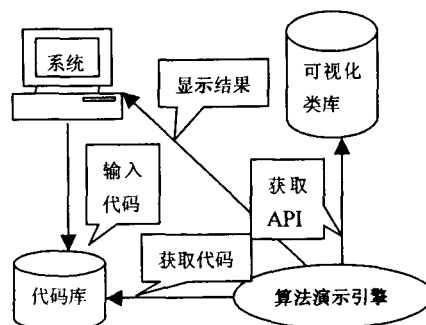


图 1 算法演示引擎 AAE 的应用例图

1.1 AAE 的结构

AAE 主要由 3 个部分组成:

- * 控制器: 用于整个算法演示过程的总体控制;
- * 解释器: 它的主要任务是对描述算法的每一行代码进行解释, 构造算法运用的数据结构;
- * 画图器: 重画对应数据结构的画图操作, 在显示区

收稿日期: 2005-10-12

作者简介: 苏莹(1980-), 女, 广东清远人, 硕士研究生, 研究方向为数据结构与算法、可视计算; 吴伟民, 副教授, 硕士研究生导师, 研究方向为数据结构与算法、可视计算。

域显示出来。

1.2 AAE 的执行过程

系统启动时,发送处理信息给 AAE 时,AAE 的控制器启动,AAE 控制器从代码库中读入算法代码,然后发送执行信息到解释器,解释器接收到信息后,逐行读取算法代码,对其进行解释。解释器对一行代码解释完毕,就发送重画信息到画图器,画图器就自动调用相应的重画方法(具体定义请参考关于数据结构可视化类库的文档),执行相应的画图操作。

2 解释算法的基本原理

对算法的执行过程进行可视化的关键是解释算法,获取进行可视化的信息。AAE 采用模拟算法执行的方法对算法进行解释。在算法演示软件中,算法是用伪码语言(如类 C 语言,类 Java 语言)描述的。伪码语言包括了变量和高级程序语言中的基本控制结构:顺序结构、选择结构和循环结构。模拟算法运行的过程就是对算法解释的过程。对算法的每一行代码,都有相应的解释方法对其进行解释。算法的每一行代码完成对某个数据结构进行的若干个操作,因此解释算法的过程就是构造某个数据结构的过程。数据结构可视化类库基本实现了对各种数据结构进行各种操作的方法(具体实现请参考关于数据结构可视化类库的文档)。算法中的变量封装在一个内部类中。

解释算法的关键问题是如何实现算法中基本的控制结构。可通过在解释方法中设置整型变量 nextStep 的值来实现。nextStep 记录的是算法代码行的行号。下面具体阐述如何实现算法的结构控制。

2.1 顺序结构

在模拟实现顺序结构时,只要顺序设置变量 nextStep 的值就可以实现顺序执行。

例如,代码行 L_0 到代码行 L_1 是一个顺序执行的过程。示例语句如下所示:

L_0 :sentence0; // L_0 表示第 0 行代码,第一行代码是语句 sentence0

L_1 :s1; // L_1 表示第 1 行代码,第二行代码是语句 s1

当解释到代码行 L_0 时,nextStep 的值为 0,执行解释方法 source0(),source0()用于解释代码行 L_0 。解释方法 source0() 的主要任务是解释 sentence0 语句和设置 nextStep 的值加 1,表示下一个执行的是代码行 L_1 。

source0() 的基本实现如下:

```
source0() { //解释代码行  $L_0$ :sentence0;
    (解释 sentence0 语句)
    nextStep ++;
}
```

2.2 循环结构

在模拟执行循环语句时,通过这个循环语句的循环条件判断来决定 nextStep 的值从而进行相应的流程控制。对于循环语句的前后花括号 {}, 后花括号 “}” 必须是独立

的代码行,当模拟程序运行到这一行,则将 nextStep 的值设置为进行循环条件的判断的代码行号,重新判断循环条件是否成立,若条件成立,进入循环体;若条件不成立,跳出循环。这样就模拟执行了循环语句。

例如,代码行 L_2 到代码行 L_4 是一个 for 语句,代码行 L_5 是执行完 for 语句后第一行代码。示例语句如下所示:

L_2 :for(e1;e2;e3) {

L_3 : s1;

L_4 :}

L_5 :s2;

分别解释代码行 L_2, L_3, L_4 的解释方法的基本实现如下:

source2() { //解释代码行 L_2 :for(e1;e2;e3) {

//判断是否第一次执行代码行 L_2 ,若是,解释表达式 e1,否则,解释表达式 e3

//布尔类型变量 first 的初始值为 true

if(first) {

(解释表达式 e1)

first = false;

} else {

(解释表达式 e3)

}

//若 for 语句中表达式 e2 值为真,设置 nextStep 的值为 3,表示下一个执行代码行为 L_3

//跳出循环,设置 nextStep 的值为 5,表示下一个执行代码行为 L_5

if(表达式 e2 值为真)

nextStep = 3;

else

nextStep = 5;

}

source3() { //解释代码行 L_3 :s1;

(解释 s1 语句)

nextStep ++;

}

source4() { //解释代码行 L_4 :}

//设置 nextStep 的值为 2,表示下一个执行代码行是 L_2 ,重新执行 for 语句

nextStep = 2;

}

while 语句的实现原理与 for 语句类似。

2.3 选择结构

在模拟执行选择语句时,通过这个选择语句的条件判断,然后通过方法的返回值来决定变量 nextStep 的值从而进行相应的流程控制。

例如,代码行 L_6 到代码行 L_9 是一个 if else 语句, L_{10} 是一个非 else 语句。示例语句如下所示:

L_6 :if(e1)

L_7 : s1;

L_8 :else

```
L9: s2;
```

```
L10: s3;
```

分别解释代码行 L₆, L₇, L₉ 的解释方法的基本实现如下:

```
source6() { //解释代码行 L6: if(e1)
```

```
if(表达式 e1 的值为真) {
```

```
    nextStep = 7; //设置 nextStep 的值为 7, 表示下一个执行的代码行是 L7
```

```
    } else {
```

```
        nextStep = 9; //设置 nextStep 的值为 9, 表示下一个执行的代码行是 L9
```

```
    }
```

```
}
source7() { //解释代码行 L7: s1;
    (解释 s1 语句)
```

```
    nextStep = 10; //设置 nextStep 的值为 10, 表示下一个执行的代码行是 L10
```

```
}
```

```
source9() { //解释代码行 L9: s2;
```

```
    (解释 s2 语句)
```

```
    nextStep = 10; //设置 nextStep 的值为 10, 表示下一个执行的代码行是 L10
```

```
}
```

3 AAE 的基本实现

基于算法演示引擎 AAE, 本工作室用 Java 语言开发了一个算法演示软件。这个软件不仅生动、形象地展示了算法的执行过程, 还具有良好的交互性, 如用户可以自行输入数据, 选择相应的算法, 就可以观察算法的具体执行过程。

下面详细阐述基于 Java 语言的 AAE 的基本实现。

3.1 解释器的基本实现

解释器的主要任务是对描述算法的每一行代码进行解释, 构造算法运用的数据结构。解释器的基本实现由抽象类 Interpreter 类实现。其中对每一行代码进行解释的方法为 proceed() 方法封装一组解释算法的解释方法。对应第 k 代码行的解释方法为 source k () , 解释方法主要完成两个任务: 解释代码和设置画图的控制参数。由于每个算法的代码行数都不同, 所以实现算法动态演示的子类都必须通过继承 Interpret 类重写 proceed() 方法。

```
class Interpreter {
```

```
    .....
```

```
    int proceed() {
```

```
        switch(nextStep) {
```

```
            case 0: source0(); break; //对第 0 号代码行进行解释执行
```

```
            case 1: source1(); break; //对第 1 代码行进行解释执行
```

```
            .....
```

```
            default: break;
```

```
        }
```

```
    } //调用 Drawer 中的 setPaintPara 方法, 设置重画的所需的参数
```

```
    后发送重画信息
```

```
    .....
```

```
    //返回下一步执行代码的行号
```

```
    return nextStep;
```

```
}
```

```
    //对应代码行 0 的解释方法
```

```
    source0() {
```

```
        //控制算法流程的变量
```

```
        int nextStep;
```

```
        .....
```

```
    }
```

3.2 画图器的基本实现

画图器由 Drawer 类实现。其中由里面的 paint 方法进行演示过程中的重画操作。每个数据结构类都有相应的 draw() 方法, 这些 draw() 方法的实现都封装在数据结构可视化类库中(具体的实现请参考关于数据结构可视化类库的文档)。

```
class Drawer extends Canvas {
```

```
    .....
```

```
    Datastructure d; //声明一个数据结构变量
```

```
    int displayMode; //声明一个整型变量表示采用的显示模式的类型
```

```
    Color color; //显示的颜色
```

```
    //设置演示的基本信息, 包括据结构类型, 显示模式, 颜色等
```

```
    setPaintPara(Datastructure d, int displayMode, Color color) {
```

```
        this.d = d;
```

```
        this.displayMode = displayMode;
```

```
        this.color = color;
```

```
    }
```

```
    paint(Graphics g) {
```

```
        //对数据结构进行重画操作
```

```
        Datastructure.draw(Canvas, g, d, displayMode, color);
```

```
    }
```

```
    .....
```

```
}
```

3.3 控制器的基本实现

运用线程对算法的执行流程进行总体控制, 因此控制器通过创建一个继承 Thread 类的 Controller 类来实现。基本实现如下:

```
class Controller extends Thread {
```

```
    .....
```

```
    run() {
```

```
        .....
```

```
        do {
```

```
            //启动解释器对相应的代码行进行解释, 并从解释器中获取下一个执行代码行号
```

```
            Interpreter.nextStep = Interpreter.proceed();
```

```
            .....
```

```
        } while(Interpreter.nextStep != -1); //若返回代码行号为 -1, 算法执行结束
```

(下转第 54 页)

将客户的婚姻状况表示为 M, S 两种。M 表示已婚, S 表示未婚。

会员卡等级分为金卡、银卡、铜卡 3 种等级。

(2) 模型结果分析如表 1 所示(下面只列举了部分数据)。

表 1 模型结果分析

条件 预测	条件 1	预测可能性 (所占比率%)	条件 2	预测可能性 (所占比率%)
金卡	A&M	22.32	A&S	3.88
	D&M	83.33	D&S	3.33
银卡	A&M	10.13	A&S	9.39
	D&M	9.72	D&S	86.67
铜卡	A&M	62.26	A&S	81.45
	D&M	4.17	D&S	6.67

客户分类预测模型的结果分析如表 1 所示:按照客户的薪水和婚姻状态作为预测的输入条件,将客户的会员卡等级作为预测的对象。例如:条件 D&M 表示年薪在 10w 以上的已婚客户,可能成为金卡客户的概率为 83.33%;可能成为银卡客户的概率为 9.72%。

(3) 模型的预测分析。

对新客户资料的数据进行分析,得出的预测分析结果如表 2 所示。

表 2 预测分析结果

年薪	婚姻状况	预测等级	概率	最大支持数
7~9 万元	已婚	铜卡	0.708745874	858
10 万以上	已婚	银卡	0.473684210	107

这个模型的预测结果为潜在价值客户的挖掘和新客户的获取方面提供了参考。例如:假设商家有很多新客户资料,但如何使商家有目的地去拓展新的客户群体?如何发现价值客户?那么此预测模型就是一个很好的参考。通过此模型,可以对新客户资料的自然属性进行分析,预测客户的价值,发掘那些最可能成为商家盈利的价

值客户。商家针对这些客户,可以采取一定的措施(如短信、赠品、积分等方式)来吸引客户,使他成为商家真正的价值客户^[5]。

这种建模的方法同样也适用于信贷风险投资中。假设客户的属性特征为输入列,以客户的信用风险度为预测对象建立客户风险预测模型,当有新的客户要申请信贷的时候,就可以使用预测模型,对客户进行信用风险度的预测,从而降低信贷风险。

3 结束语

文中主要讨论了利用微软的 MS Analysis Server 分析工具、.NET 开发平台和微软的决策树算法,通过对客户自然属性的分析,以会员卡等级作为预测对象来建立客户分类预测模型过程和方法,特别是通过对模型训练所得出的结果在客户的潜在价值预测和新客户的发掘方面具有一定的实用价值。

微软的工具在商业项目的开发方面与其他工具相比具有操作方便、开发周期短等优点。不足的是微软的分析工具仅提供了决策树算法和聚类二种数据分析算法,使其应用面受到一定的限制。但对于设计一般要求的分析系统而言,它不失为一种高效、快捷的好方法。

参考文献:

- [1] 张学军,吴 潇.CRM 实施宝典[M].北京:国防工业出版社,2005.
- [2] 贾 琳,李 明.基于数据挖掘的电信客户流失模型的建立与实现[J].计算机工程与应用,2004(4):185-187.
- [3] Gunderloy M. SQL Server 开发指南(联机分析处理)[M].张 伟,宋 霞译.北京:电子工业出版社,2001.
- [4] Seidman C. SQL Server 2000 数据挖掘技术指南[M].刘 艺,王鲁军等译.北京:机械工业出版社,2002.
- [5] 史威福特 R S. 客户关系管理[M].杨东龙等译.北京:中国经济出版社,2001.

(上接第 51 页)

而且也对实现其他动态演示系统起一定的启发作用。

参考文献:

- [1] 吴伟民.数据结构和算法的可视化教学研究与实践[J].现代计算机,1999(3):35-37.
- [2] 林桂伍,黄 峰,陈峥杉,等.可视化环境下 CAI 课件系统设计与实现[J].计算机应用研究,1999(4):106-107.
- [3] 高 翔.用 Java 多线程实现数据结构算法动态演示[J].北京联合大学学报,2002,16(2):63-66.
- [4] 吴志刚,韩瑶香,苏安婕.《数据结构》多媒体 CAI 系统的研制[J].郑州纺织工学院学报,2001,12(4):46-49.
- [5] 彭玉青,肖国玺,杨 昕.数据结构算法动态演示 CAI 软件的实现[J].河北工业大学学报,2000,15(1):1-4.

4 结束语

实践证明,通过构造算法演示引擎 AAE,可以实现交互式的动态演示。该算法演示引擎已在本工作室开发的算法演示软件中应用,取得良好的效果。但是,在现阶段,AAE 的智能性还有待提高。针对 AAE 的不足之处,未来的工作就是实现一个通用的算法演示引擎。通用的算法演示引擎可以自动地生成解释方法,这些解释方法可以对具有一定规范的伪码语言进行解释。这样,只需要实现算法演示引擎提供的接口,就可以实现动态演示。构造算法演示引擎的思想不仅为实现算法的动态演示提供了思路,