文章编号:1001 - 9081(2009)10 - 2865 - 04

基于 SVG 的专用公式编辑工具的设计与实现

唐 勇,吴尽昭,陈剑锋
(中国科学院 成都计算机应用研究所,成都 610041)
(idlerty@ gmail. com)

摘 要:提出了一种基于 SVG 技术的专业领域公式编辑工具解决方案。使用自定义的公式输入规则、Java 图形编程技术和 Batik 软件包设计并实现了一个应用于交互式马尔可夫链(IMC)的并发系统性能评价系统的公式编辑器。该方案设计完善、操作简单、编辑方便、公式显示效果出色、通用性强,能方便地进行转化以适应于多种专业领域。

关键词:公式编辑工具;公式显示;可伸缩矢量图形;Java 图形编程;Batik
中图分类号: TP311.1 **文献标志码**:A

Design and implementation of SVG-based specific formula editing tool

TANG Yong , WU Jin-zhao, CHEN Jian-feng
(*Chengdu Institute of Computer Application, Chinese Academy of Sciences, Chengdu Sichuan* 610041, *China*)

Abstract: The SVG-based formula editing tool used for specific domain solutions was proposed. With customized formula input rules, Java graphics programming and Batik software package, the editing tool was designed and implemented for evaluating performance of concurrent system described by Interactive Markov Chains (IMC). The solution has a complete structure, easy to use GUI, convenient editing functions, etc. It displays formula perfectly and holds strong compatibility to be widely employed in various areas.

Key words: formula editing tool; formula show; Scalable Vector Graphic (SVG); Java graphics programming; Batik

0 引言

目前流行的数学计算软件或数学排版软件或是通过定义一系列词法来规定用户输入,或是使用所见即所得式的编辑按钮方便用户输入。除了这些区别,它们都是将用户输入转化成一定的图片格式(如 png 格式)提供给用户使用。这种方式使得在公式编辑完成后的修改工作具有一定的难度。此外,在需要重用这些公式的场合,往往需要用户再次以手工方式重新输入编辑。在各个专业领域中,往往需要编辑显示大量的复杂公式,而通过普通的数学计算软件或排版软件由于存在前述不便经常无法完全满足要求,由此产生了开发基于各专业领域的公式编辑工具的需求。实践表明,由于各专业领域公式的复杂性和公式元素的多样性,无法在软件界面中使用 ASCII 字符,只能使用一种可定制的图形规范来实现公式的编辑及显示。

本文针对基于交互式马尔可夫链(Interactive Markov Chains,IMC)的并发系统性能评价领域,从构造一个实际的性能评价系统的需求出发,通过研究用来刻画该种并发系统的一种时序逻辑 aCSL,给出了一种切实可行的 aCSL 逻辑公式编辑工具的解决方案。该方案具有词法定义简洁、软件操作简单、公式输入快捷、公式显示效果出色、公式编辑修改方便等优点。稍加改造,可构建其他多种需要编辑显示复杂公式的专业领域公式编辑软件。

1 预备知识

基于软件可移植性及与已有技术接口兼容性的考虑,本文中给出的解决方案拟采用 Java 技术来实现图形用户界面,下面给出其他知识的相关介绍。

1.1 aCSL 逻辑公式

交互式马尔科夫链(IMC)是一个功能与性能混合的并发系统模型,它提供了一个完美的可组合化的性能评价框架。aCSL(Action-based CSL)是一种新提出的针对 IMC 模型特点的逻辑,它是基于动作的一个时序逻辑[1]。aCSL 逻辑的语法表述如下:

对 $t \in R_{>0}$, $A,B \subseteq$ 动作集合 Act,aCSL 的路径公式由以下语法产生:

$$\varphi ::= \Phi_A \bigcup\nolimits_B^{<t} \Phi \mid \Phi_A U \bigcup^{<t} \Phi$$

本文将从上述公式出发,通过分析其结构特点,采取一系列分解手段达到在软件界面中编辑并显示这种公式的目的。

1.2 SVG

SVG(Scalable Vector Graphics)是互联网联盟(W3C)公布推荐的一种基于 XML 的二维图形描述语言标准,目的在于满足应用程序中日益增长的对动态、可缩放和平台无关地展现复杂内容并实现灵活交互的需求,具有强大的可重用性和伸缩性。SVG 严格遵从 XML 语法,并用文本格式的描述性语言来描述图像内容,因此是一种和图像分辨率无关的矢量图形格式。SVG 图形格式具有以下优点:1)图像文件可读,易于修改和编辑;2)与现有技术可以互动融合;3)SVG 图形格式可以用来动态生成图形。例如,可用 SVG 动态生成具有交互功能的图片,嵌入应用程序中,并显示给终端用户。本文中将对 aCSL 逻辑公式进行分解,利用 SVG 易于修改和编辑且能动态生成的优点来储存公式。

1.3 Batik

Batik 是 Apache 软件基金会(Apache Software Foundation)

收稿日期:2009 - 04 - 08 **基金项目**:国家 863 计划项目(2007AA01Z143);国家 973 计划项目(2007CB310803)。
作者简介:唐勇(1982 -),男,安徽庐江人,硕士研究生,主要研究方向:形式化方法; 吴尽昭(1966 -),男,黑龙江齐齐哈尔人,研究员,博士生导师,主要研究方向:形式化方法,自动推理、验证技术; 陈剑锋(1983 -),男,新疆乌鲁木齐人,博士研究生,主要研究方向:形式化方法。

开发的一个开源项目。该项目的目标是提供一组核心模块，通过使用这些模块可以实现特定的SVG解决方案。Batik基于Java技术，其功能包括图像的生成、显示和操作。

本文将使用Batik在软件界面中显示动态生成的SVG格式储存的性质公式。

2 设计与实现

本文的目的是从实现一个基于IMC的并发系统的性能评价系统的实际需求出发，设计并实现一个并发系统性质编辑器，以作为性能评价系统的一个单独功能模块。

2.1 性质编辑器设计

2.1.1 设计思路

通过对aCSL逻辑语法进行细致分析，可以对用aCSL逻辑语法表述的公式进行分解，采取化整为零的方法将一个在软件界面中无法用程序设计语言直接描绘的公式分解成多个单独的部分，从而方便地实现公式中各个参数的存储以及公式在软件界面中的显示。在这种思路的指导下，本文构建的性质编辑器将把用户以规定格式输入的性质公式进行分解，分解成多个公式元素后，参照规定的各种公式元素的配置参数，生成SVG格式的文件来储存公式并使用Batik软件包在性质编辑器的软件界面中显示公式，给用户以直观的反馈。

以如下aCSL公式形式为例：

$$\varphi_1 = \Phi_{1\ A}\bigcup^{\Delta t}_{B}\Phi_2 \tag{1}$$

式(1)的直观含义是并发系统中的一条最终到达Φ_2状态的路径。假设欲对某并发系统进行性能评价，性能评价人员需要设定待验证的性质的形式如式(2)所示，其含义是针对此并发系统需要验证的第n条性质是执行路径φ_1的概率满足Δx。

$$\Phi_N = P_{\Delta X}(\text{true}\ _A\bigcup^{\Delta t}_{B}\Phi_2) \tag{2}$$

其中：N是自然数，是性质的编号；$x,t \in R$；$\Delta \in [\leq, <, \geq, >]$；$A,B \subseteq$动作集合，或为空集，或为“$\{F_1\}$”、“$\{F_2,F_5\}$”之形式。结合式(2)中各个部分的含义，不难将式(2)的结构分解成如表1所示各个元素。

表1 性质公式分解对应表

公式元素	元素名	公式元素	元素名
Φ	性质名	Φ_2	目的状态(集合)
N	性质编号	A	前置动作(集合)
=	等号	B	后置动作(集合)
P	概率标识	U	*Until*标识
Δx	概率约束条件	Δt	时间约束条件
x	概率约束值	t	时间约束值
Φ_1	起始状态(集合)	Δ	概率和时间约束符
(,)	圆括号	$n \in N$	动作编号

2.1.2 功能设计

本文所设计的性质编辑器的用途，是作为一个并发系统性能评价软件的独立模块存在的性质公式编辑器。通过这个编辑器，用户能够方便地输入所需验证的并发系统性质，且编辑器应该提供给用户一个关于此性质的直观表示，以方便用户检验输入的正确性。基于这样的设计目标，结合2.1.1小节的设计思路，下面给出性质编辑器的功能结构如图1所示。

在本文实现的性质编辑器中，所需验证的公式由用户按照既定词法将公式以字符串方式输入性质公式编辑模块。性质公式编辑模块对用户输入字符串进行词法检查，若不符合既定词法，则给用户以提示，否则转入性质公式解析模块。性质公式解析模块对公式字符串按照既定词法进行解析，从中分解出性质公式的各个公式元素，将解析出的公式元素按照性能评价系统的后台定义数据结构加以封装，同时把解析出的每个公式元素内容结合既定的各种参数(如元素类型、元素长度等)组合成SVG格式文件存储在本地文件系统，并激活性质公式显示模块。显示模块调用SVG文件并在软件界面中图形化显示SVG文件所描述的公式，给用户直观体验。在用户检查无误后，将以后台数据结构封装好的数据提供给性能评价系统中相应模块，结束本次公式编辑过程。

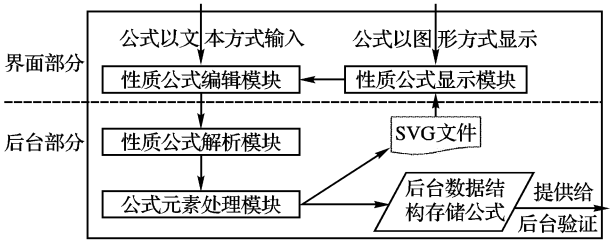


图1 性质编辑器功能结构

2.1.3 性质显示格式设计

由于在SVG中显示的图形都是由图元组成的，因此需要根据图元来定义性质公式元素的显示格式以及各个元素的坐标参数。图2所示是在以交互式马尔科夫链所刻画的并发系统中一个具有代表性的待验证性质公式。表2是从图2出发，总结归纳了所有待验证性质的结构特征后，定义的性质公式每种元素的显示格式及坐标参数。

$$\varphi_2 = P_{>0.1}(\text{true}_{\{F_2\}}U^{<5}_{\{F_3,F_5\}}\{\Phi_2\})$$

图2 一个典型的性质公式(以MathType绘制)

表2 公式元素的显示格式及坐标参数

元素名	字号	纵坐标/像素
性质名	20	50
性质编号	12	52
等号	20	50
概率标识	20	50
概率约束值	12	52
圆括号	20	50
起始状态(集合)	20	50
目的状态(集合)	20	50
前置动作(集合)	12	52
后置动作(集合)	12	52
Until标识	28	50
时间约束值	12	36
动作编号	8	56

表2中没有给出每种元素的横坐标，这是因为元素的横坐标是由待验证性质公式的动态性决定的。对于同一个并发系统，不同的用户存在着不同的性能需求，基于不同的需求，就会产生不同的待验证性质公式。如某验证人员所设定的性质公式的前置动作集合中只包含元素F_2，其他验证人员需要将前置动作集合设定为$\{F_2,F_4\}$，如此，两个性质公式在显示时对于紧邻前置动作集合的Until标识的横坐标要求就不一致。性质公式的改变，即便是很小的变动，都会影响公式在显示时的效果。图2中显示的公式是直观而紧凑的，如果采用规定横坐标的策略，则或者松散，或者重叠(如图3所示)，或者兼有松散和重叠而较难得到如图2的效果。从表2可以看出，

公式中的各个字符的字号不尽相同,由字号的不同,又引起每个字符占用的长度不同,比如 Until 标识(图 2 中的“U”型符号)占用的长度比前置动作集合中动作的编号(图 2 中“F_2”的“2”)长。因此,为了保证在最终产生的公式图片中不会有不同元素重叠显示,有必要规定每种公式元素占据的显示长度,以确保在编辑器运行时能根据对性质公式各元素的分解结果和既定的元素显示长度正确计算出元素的横坐标。通过对 aCSL 公式结构的分析,统计出 aCSL 公式中常用字符如下所示。

1)10 个阿拉伯数字:0,1,2,3,4,5,6,7,8,9;

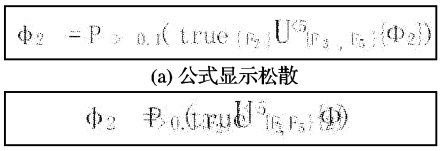
2)52 个大小写英文字母;

3)部分希腊字母:Φ,φ;

4) 部分数学符号:≤,≤,<,≥,>,∧,~,Δ;

5)部分标点符号:逗号(“,”),中括号(“[,]”),圆括号(“(,)”),大括号(“{,}”)。

表 3 是经过不断尝试调整得到的每种元素内容占用的显示长度表。



(a) 公式显示松散

(b) 公式显示重叠

图 3　公式显示松散和公式显示重叠

表 3　元素占用长度表

元素名	占用长度/像素
性质名	20
性质编号	15
其他上下标	7
概率标识	20
Until 标识	20
=	20
≤,≥	20
{, }	7
逗号	5
其他单个字符	10
true	50
false	60

通过上述公式元素显示格式及占用长度的定义,结合对公式进行分解而得的元素结果集,可以得到结果集中每种元素的横坐标。本文定义的计算方法如下:

若该元素为结果集的第一个元素,则该元素横坐标为 5(单位:像素),否则该元素横坐标 = 前一元素横坐标 + 前一元素占用(总)长度。

其中,元素占用总长度针对由单个字符组合而成的元素,是由单个字符占用长度(表 3 中元素名为“其他单个字符”)乘以字符数目得到。如某性质的编号为“ab31”,则其占用总长度 = 10 × 4 = 40(像素)。

2.1.4　性质公式词法定义

为了便于用户输入及公式解析,我们定义了一套输入规则,通过这些规则,用户可以方便地将类似图 2 显示的公式以字符串形式输入编辑器。规则中定义了一系列以“$”开头的关键字,对应标识公式中的每种元素。如表 4 所示。此外,系统尚有如下规定。

1)元素的子元素在一对方括号(“[ ]”)内描述,如图 2 所示公式的性质及其编号表示为 $ BPHI $ INX[2];

2)由多个元素组合成一个元素时,多个元素之间以“,”分割。如图 2 所示公式的概率及其约束条件表示为 $ PROB $ INX[>,0.1],Until 标识及其时间约束条件表示为 $ UNTIL $ SUPS[<,5]。

表 4　系统中定义的关键字

关键字	对应元素
$ INX	下标
$ SUPS	时间约束条件
$ PROB	概率标示
$ START	开始状态(集合)
$ DEST	目的状态(集合)
$ PRE	前置动作(集合)
$ POST	后置动作(集合)
$ UNTIL	Until 标识
$ CONJ	合取符号(∧)
$ NOT	逻辑非符号(~)
$ EMPTY	空集符号(∅)
$ BPHI	希腊字母 Φ
$ LPHI	希腊字母 φ

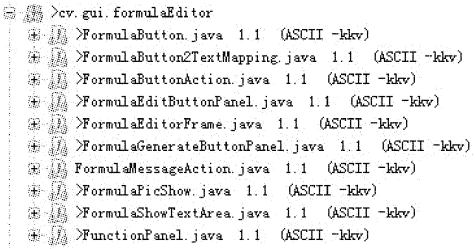
如此,图 2 所示公式可以用如下字符串表示。

$ LPHI $ INX[2] = $ PROB $ INX[>,0.1]($ START[true] $ PRE[F $ INX[2]] $ UNTIL $ SUPS[<,5] $ POST[F $ INX[3],F $ INX[5]] $ DEST[$ BPHI $ INX[2]])　　(3)

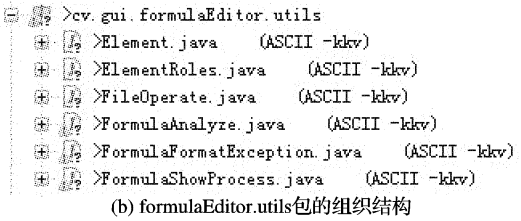
2.2　性质编辑器实现

通过上述设计阶段,本文实现构造性质编辑器的必备条件。首先是系统结构设计,其次是公式显示格式设计,最后是公式输入词法定义。在这三部分工作的基础上,本文着手实现了一个基于 IMC 的并发系统的性质编辑器。本编辑器在 Windows 操作系统下使用集成开发环境 Eclipse 开发。

开发完成的代码主要由两个 Java 包构成:formulaEditor 和 formulaEditor.utils。其中,formulaEditor 包主要用于构建性质编辑器的图形用户界面,formulaEditor.utils 包主要完成解析用户输入的性质公式、创建 SVG 文件以及在图形用户界面中显示公式的功能。两个 Java 包的组织结构如图 4 所示。



(a) formulaEditor包的组织结构



(b) formulaEditor.utils包的组织结构

图 4　formulaEditor 包和 formulaEditor.utils 的组织结构

图 5 是开发完成后的软件界面的效果演示,主要由五部分组成。

功能按钮区:提供新建性质公式、打开已有性质公式、保

存性质公式和退出编辑器四项主要功能。

性质列表区:保存已经编辑完毕的性质。

编辑按钮区:提供一组形象化按钮以方便用户输入,用户不再需要准确记忆软件设定的词法系统。

公式编辑区:此处显示以字符串形式描述的性质公式,用户可以通过使用编辑按钮配以必要数据的方式或使用纯手工方式在此处编辑修改公式。

公式显示区:用户在输入或编辑公式字符串后,点击“预览”按钮,公式显示区会将公式直观显示出来,以方便用户核查。用户核查完毕后,点击“产生”按钮,此公式将被创建并添加到性质列表,同时以规定数据结构封装好的公式会传递给后台进行后续工作。

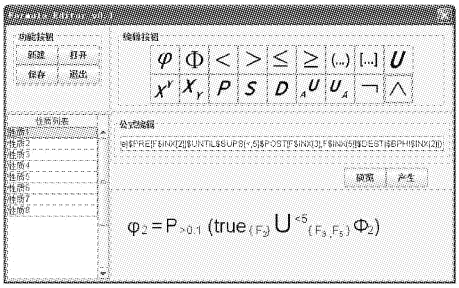


图5 本文实现的性质公式编辑器界面

图5中显示的公式和图2是同一公式。其中图5是以式(3)作为输入,由性质编辑器解析、创建并生成显示,图2是以MathType绘制。对比二者,不难看出由本文构造的性质编辑器的显示效果丝毫不逊色于MathType。下面列出的代码片段是本编辑器运行过程中创建的SVG文件,对应图5显示的公式。

代码片段1(对应图2公式的SVG文件内容结语):

```
<?xml version = "1.0" encoding = "UTF - 8"? >
< svg xmlns = "http://www. w3. org/2000/svg" >
< text x = "5" y = "50" style = "font - size: 20pt; " > &#x3C6; </
text > &#x3C6;
< text x = "25" y = "52" style = "font-size: 12pt; " >2 </text >2
< text x = "40" y = "50" style = "font-size: 20pt; " > = </text > =
< text x = "60" y = "50" style = "font-size: 20pt; " > P </text > P
< text x = "80" y = "52" style = "font-size: 12pt; " > &gt; </text >
&gt;
< text x = "90" y = "52" style = "font-size: 12pt; " >0. 1 </text >0.
1
< text x = "120" y = "50" style = "font-size: 20pt; " >( </text >(
< text x = "130" y = "50" style = "font-size: 20pt; " > true </text >
true
< text x = "180" y = "52" style = "font-size: 12pt; " >{ </text >{
< text x = "190" y = "52" style = "font-size: 12pt; " > F </text > F
< text x = "200" y = "56" style = "font-size: 8pt; " >2 </text >2
< text x = "205" y = "52" style = "font-size: 12pt; " >} </text >}
< text x = "215" y = "50" style = "font-size: 28pt; " > U </text > U
< text x = "245" y = "36" style = "font-size: 12pt; " > &lt; </text >
&lt;
< text x = "255" y = "36" style = "font-size: 12pt; " >5 </text >5
< text x = "265" y = "52" style = "font-size: 12pt; " >{ </text >{
< text x = "275" y = "52" style = "font-size: 12pt; " > F </text > F
< text x = "285" y = "56" style = "font-size: 8pt; " >3 </text >3
< text x = "295" y = "56" style = "font-size: 12pt; " >, </text >,
< text x = "300" y = "52" style = "font-size: 12pt; " > F </text > F
< text x = "310" y = "56" style = "font-size: 8pt; " >5 </text >5
< text x = "320" y = "52" style = "font-size: 12pt; " >} </text >}
< text x = "330" y = "50" style = "font-size: 20pt; " > &#x3A6; </
text > &#x3A6;
< text x = "350" y = "52" style = "font-size: 12pt; " >2 </text >2
< text x = "360" y = "50" style = "font-size: 20pt; " >) </text >)
</svg >
```

3 结语

本文从构造一个针对基于IMC的并发系统的性能评价工具的实际需求出发,设计并实现了一个词法定义简单、用户输入方便、显示效果出色的性质编辑器。该编辑器使用了移植性很强的Java语言编写,实现了平台无关性,达到了一次编写,随处运行。另外采用了SVG这一基于XML的二维图形描述标准,文本格式的公式存储方式使得公式的编辑易于进行。

尽管本文提出的方案有着很多显而易见的优点,但是囿于需求和时间的限制,本文只基本实现了满足作为性能评价工具子模块的功能。在进一步的工作中,作者将首先着手于完善从预览出的公式直接点击即可进行公式修改的功能,进而简化软件的词法设计,争取实现一个无词法限制,所见即所得的公式编辑器。

参考文献:

[1] 覃广平. 交互式马尔可夫链: 理论与应用[D]. 北京: 中国科学院研究生院, 2006.

[2] 黄凯伟. SVG开发实践[M]. 北京: 电子工业出版社, 2008

[3] HORSTMANN G S. CORNELL G. Core Java 2 Volume I, Fundamentals[M]. 6 ed. [S. l.]: Prentice Hall, 2000.

[4] ECKEL B. Thinking in Java[M]. [S. l.]: Prentice Hall, 1998.

[5] KOLESNIKOV A. Java drawing with apache batik: A tutorial[M]. [S. l.]: Brainy Software, 2007.

[6] KNUTSCH K F. Java用户界面编程指南[M]. 北京: 电子工业出版社, 2002.

(上接第2864页)

[4] JOE L, ROGER L. Multiple vehicle routing with time and capacity constrains using genetic algorithms [C]// Proceedings of the 15th International Conference on Genetic Algorithms. San Francisco: Morgan Kaufmann Publishers, 1993: 452 - 459.

[5] BAKER B, YECHEW A. A genetic algorithm for the vehicle routing problem[J]. Computers Operations Research, 2003, 30(2): 787 - 800.

[6] 谢秉磊, 李军, 郭耀煌. 有时间窗的非满载车辆调度问题的遗传算法[J]. 系统工程学报, 2000, 15(3): 29.

[7] 郎茂祥, 胡思继. 用混合遗传算法求解物流配送路径优化问题的研究[J]. 中国管理科学, 2002, 10(5): 51 - 52.

[8] 杨宇栋, 朗茂祥, 胡思继. 有时间窗车辆路径问题的模型及其改进模拟退火算法研究[J]. 管理工程学报, 2006, 20(3): 104 - 107.

[9] HOLLAND J H. Adaptation in natural and artificial systems[M]. Cambridge, MA, USA: MIT Press, 1975.

[10] 陈国良, 王煦法, 庄镇全等. 遗传算法及其应用[M]. 北京: 人民邮电出版社, 1996: 80.

[11] 蒋昀昕. 自适应小生境遗传算法的研究[D]. 淮南: 安徽理工大学, 2008: 19.

[12] 王小平, 曹立明. 遗传算法——理论、应用与软件实现[M]. 西安: 西安交通大学出版社, 2002: 75.

[13] 林清国. 基于混和遗传算法的有时间窗车辆路径问题研究[D]. 济南: 山东大学, 2007: 26.