可视化编程中识别计算思维

刘小燕 陈艳丽
(河南理工大学计算机科学与技术学院 焦作 454003)

摘 要 计算思维已成为教育研究的流行词。可视化编程可使越来越多的学生学习计算思维。先前的研究大多集中于可视化编程产生的动机水平,并未研究学生从可视化编程中实际学到了什么样的计算思维。根据程序行为相似性语义分析学生使用可视化语言创建的游戏和模拟,提出了可视化自动评价方法,即计算思维模式图,来评价学生创建游戏和科学模拟中使用的计算思维模式,同时也指出学生从游戏设计到科学模拟中计算思维的转移。

关键词 计算思维,可视化编程,计算思维模式,计算思维识别
中图法分类号 TP312 **文献标识码** A

Recognition of Computational Thinking in Visual Programming

LIU Xiao-yan CHEN Yan-li
(College of Computer Science and Technology, Henan Polytechnic University, Jiaozuo 454003, China)

Abstract Computational thinking has become popular in education research. More students can learn computational thinking in visual programming. Previous studies center on the motivation level that visual programming produces, but don't focus on what kinds of computational thinking students learn from visual programming. This paper made semantic analysis on games and simulations that students created by using visual language on the basis of program behavior similarity, brought up an automatic and visual assessment method that is computational thinking pattern graph. This method can evaluate computational thinking pattern used in games and scientific simulations created by students, at the same time, points out students' possible transfer of computational thinking from game design to scientific simulations.

Keywords Computational thinking, Visual programming, Computational thinking pattern, Recognition of computational thinking

1 简介

可视化编程语言在教会学生如何编程上变得越来越有效。可视化编程至少能够减少语法问题,例如不仅仅是一个分号导致整个程序出错。可视化编程的教育目的主要是动机。可视化编程是一种拖放功能风格编程,主要面向终端用户开发,使编程变得更简单。可视化编程包括游戏设计,动机水平可以非常高,支持计算机科学学习。但是问题出现了,学生实际学了什么。例如,由于可扩展游戏设计课程嵌入到已有的必修课程里面,大量的在校生学习 AgentSheets 等创作工具,而以前这些必修课程包括文字处理、键盘操作等教学应用[1]。这些课程要求大多数学生参与游戏设计模块。令人惊讶的是,即使强迫曝光,动机也非常高。例如,76%的女生和67%的男生说他们要继续学习游戏设计课程。这些探索动机大小的项目得到了资助机构很好的支持,包括国家自然科学基金的支持。然而,这些调查范围都较小,学生实际通过使用可视化编程语言来学习计算机科学并不是很清楚。

计算思维在教育研究领域越来越流行[2]。许多高校教育工作者反馈,他们听过术语"计算思维",但是不理解它是什么。尽管教育工作者没有"计算思维"统一的定义,但是他们都期望计算思维产生影响[3]。在计算思维教育中,教育者愿意教授游戏设计类课程,向学生提问"现在,你可以开始'俄罗斯方块'游戏,你能编写一个科学的模拟程序么?"这是个大胆的问题,暗含了雄心勃勃的目标,包括一个可测试条件。如果所有的学生学习简化的可视化编程语言只是为了游戏而不是应用更一般技能的能力像计算模拟,那么计算思维就不会产生。不管计算思维是什么,教育者都应当能让学生将计算能力应用到解决各种各样的问题中,这才是真正的实验[4]。

为了评价计算思维,我们需要能够识别各种各样的计算思维的技能。现有的许多尝试实质上大多局限于语法层面的技能研究。一些研究探讨可视化编程语言相对于文本语言的优势。在句法层面,这些比较是非常困难的,因为可视化和文本语言不能很好地匹配。少量研究发现了类似性语言,像 Logo 和 Scratch,它们各有优势,但并未找到可视化语言和学习技能之间存在的简单的因果关系。句法分析具有内在局限性,局限于程序的形式(语法),而不是意义(语义)。这主要不是因为人们对语义不感兴趣,而是因为从现有的程序中推断语义本质上很困难。语义程序分析领域涉及广泛,包括逆向

本文受国家自然科学基金面向社区的协同检索方法研究(F020511)资助。
刘小燕(1981—),女,硕士,讲师,主要研究方向为智能信息处理,E-mail:xyanliu@hpu.edu.cn;陈艳丽(1981—),女,硕士,讲师,主要研究方向为计算机测控。

工程和设计意图恢复。尽管限制一定范围，比如编译器具有能检测并行运行的代码而不是顺序运行的代码的能力，语义发现依然很难。在教学环境中，教育者往往根据评估标准做出最佳判断，评估标准为所需的游戏提供检查清单。比如，斯坦福大学教育学院开发了一种基于准则的方法，为中学生使用 AgentSheets 建立游戏评分。这些检查清单对教师很有帮助。教师运行学生游戏，来检查游戏应该表现出的行为。例如，在青蛙游戏中，用户应该能够使用光标键控制青蛙的位置，这是评价准则的一点。然而，以准则为基础的方法效果不好，如果可能，对于可扩充编程作业，学生可以开发几乎无限制范围的游戏或者模拟设计。

本文描述了一个自动识别方法，它基于“程序行为相似性”语义分析识别计算思维模式。该方法可以以一定的概率识别没有程序运行的语义层模式的存在。本文将介绍计算思维模式的定义、描述识别这些模式的方法、显示一些从中学到大学级别的游戏和模拟例子。本文的目的是以独立于应用领域的方式开发一种框架来识别计算思维能力。这种想法对可视化编程语言学习来说非常重要，因为它适合于以人为中心的计算。例如，自动检测计算思维模式的能力使得可视化编程语言课程适应个别学生。理想情况下，我们将证明这些模式不仅存在于许多不同的应用领域，而且我们将以一种可测试转移的方式来教这些模式，从而开始对这个问题得到一些启示，“现在你可以开始编写俄罗斯方块游戏，你能编写科学模拟程序么?”。

2 语义程序识别方法

程序识别可以根据语法或者语义进行，前人对此做了很多研究，提出了不同的程序识别方法，比如基于抽象模式[5]、基于文本[6]、基于度量值[7]、基于结构[8]、基于图的程序识别方法[9]。这些方法主要用来识别程序的相似性，但是也都存在一定的缺点。例如基于抽象模式，需要人工建立知识库和规则，不能自动识别程序；基于文本的方法，则忽略了程序的语法和语义，不能识别代码多样化等。虽然不同的方法能在一定程度上识别程序相似性，但是识别在该程序中的计算思维模式，以便于学习者据此改变自己编写程序时的理念或者习惯，以更有利于程序的优化，这也是本文的研究重点。

以可视化编程为基础的创作工具被创建和应用在许多应用领域，特别是在游戏设计、计算科学以及机器人学。图 1 描绘了这些应用领域以及随着时间的推移这些技术的转移过程。图 1 体现了 4 点：(1)一系列指定通用对象交互的计算思维模式，这些通用对象可以在许多领域中找到，包括游戏设计、计算科学和机器人学；(2)同层次的纵轴代表介绍和连接这些概念的迭代方法，例如游戏设计中的随机运动概念上类似于物理学中的布朗运动；(3)从简单的计算思维模式中定制，比如物体高速前进的碰撞类似于马斯洛的需要层次理论，这些概念在图中基于彼此相互建立；(4)机器人学、计算科学和游戏设计这 3 大计算机科学领域的连通性增强。

理想情况下，学习从简单概念开始，逐步进展到更复杂的概念。图 1 说明学习迁移过程能通过使用它们建立的相互结合的计算思维模式进行检测。根据经验，在本学期的各种类型的游戏设计课程中，增加了计算思维的复杂性及其在学生中转移的合理证据。总体来说，图 1 深刻地描绘了计算思维转移过程，就像学生使用渐增的复杂模式组合。

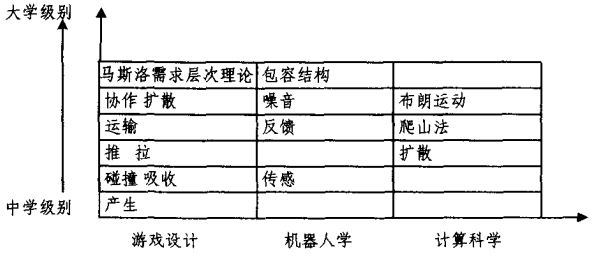


图 1 计算思维演变

本研究的目的是开发一个自适应机制来支持可视化编程教学。识别计算思维能力是计算技能评估的第一步，并以迫切需要的方式来解决游戏设计、计算科学和机器学应用领域的挑战。采用流的概念，一个以人为中心的方法能够以必要技术平衡设计挑战来适应用户的特定需求。这将使初学者逐渐学会编程技巧，从简单的应用开始，例如基本的游戏设计，然后朝向更复杂的游戏和计算科学模拟。

2.1 程序行为的相似性

我们收集了数千个中学生开发的游戏，同时从大学本科生和研究生游戏设计课程教学中收集了更复杂的游戏和科学模拟，并且完成了评估学习等不同项目中需要量化的技能超越动机的数据的收集。例如，能够自动识别图 1 所概述的计算思维能力，会使学生和教师获得直接反馈给学生的技能水平和学习的概念。

本文根据语法和语义格式，试图制定一个能够识别学生创建的游戏中增加的模式组合复杂性的评估工具。句法评价通常集中在计算形式或结构。这种类型的评价似乎不足以显示计算思维能力的潜在增长，这将导致转移。语义评估工具可以比较学期开始创建的简单的游戏项目和学期末创建的游戏项目，该工具可以揭示学习迁移。当学生开始利用游戏设计课程获得的知识进行科学模拟时，该评估工具能显示这些游戏中转移的存在。有学家者对此进行研究，如 Lewis[10] 比较了两种用于学生游戏制作的可视化编程语言(Scratch and Logo)。该研究大多集中于动机问题来确定两个软件程序有何不同，只利用个人编程作品的知识来比较两个程序，而不考虑编程的大环境。这种研究评估基于一个纯粹的语法水平，可能显示个别概念知识，但不评价学生在多个场合使用知识的能力。句法评价对检测高层次的计算思维知识或学习迁移没有用。在这方面，语义评估工具能更准确地指示学习知识的转移，是非常有用的。

一种方法是以更多语义级别来比较和概述代码，而不是只计算程序元素(例如循环)，是为了寻找能指示程序意义的更高级别的模式。类似方法称为潜在语义分析(LSA)，是通过比较文本寻找自然语言中存在的语义信息[11,12]。计算机语言包括可视化语言，有同样的思路。就像自然语言一样，计算机语言以由语法结构构成的语句概念为基础。一方面，对计算机语言的处理应当简单，因为它们的句法规则往往更不规则。在 LSA 中堵塞是一个最基本的问题，这与计算机语言无关，因为动词的词形变化不存在。计算机语言的基元和功能比较简单。另一方面，这里所描述的方法分享了一些 LSA 的记载缺点，如“词袋”问题，来防止基于词序上的语义识别。

本文首次尝试语义类型评价，使用不同 AgentSheets 游戏中的规则来开发这些游戏的配置文件，利用高维余弦值计

算程序行为相似性。几何中夹角余弦可用来衡量两个向量方向的差异，机器学习中可借用这一概念来衡量样本向量之间的差异。根据该方法，如果单位向量正交，余弦值为0；如果它们方向相同，余弦值为1。高维余弦值越大，则游戏相似度越高。AgentSheets程序包括用户创建的“代理”，这是游戏中的人物。例如在青蛙游戏中，用户创建不同的代理，如青蛙、卡车、街道等。AgentSheets中的每个代理有描述信息指定代理人在特定情况下表现出来的行为和样子。AgentSheets中的所有行为都用“If/Then”条件语句实现。AgentSheets允许使用16种不同条件和23种不同行为的组合来创建任何给定代理的行为。在青蛙游戏中，例如，为使青蛙朝4个不同的方向移动，使用涉及4个方向的4个按键条件，每个方向一个。因此，为了使青蛙动起来，学生将编程“如果键盘向上键被击，青蛙则向上移动”。使用23个条件和16种行为，可用向量长度39表示每个游戏，其中每个矢量元素代表每个单独的条件和操作用于实现给定的游戏。使用这些向量，AgentSheets中创建的任何游戏可以与通过式(1)中所描绘的高维余弦计算相似性的任何其他游戏进行比较。

$$PBS(u,v)=\frac{\sum_{i=1}^{i=n}u_i v_i}{\sqrt{\sum_{i=1}^{n}u_i^2}\cdot\sqrt{\sum_{i=1}^{n}v_i^2}} \tag{1}$$

式(1)允许以使用的规则为基础对每个游戏做一个简单的比较。游戏的高维余弦相似性比较表明，在两个游戏有相同比例的规则但规则的数量不同时其具有较高的鲁棒性。在这种情况下，句法分析将游戏分为不同类。这使得高维余弦的使用更接近于语义类型评价，纯粹语法评价更少。因此，如果两个游戏使用完全相同比例的两个彼此完全相同的规则集或规则，那么两个游戏的相似性得分会相同。也就是说，如果使用完全不同的规则，两个游戏的相似性就为0。例如，两个正常编写的蜈蚣游戏应当有比较接近的相似性得分。为此，比较了两个经典的蜈蚣游戏的实现，它们看起来相似，两个游戏相似性得分为0.87。相似性得分分析给我们一个初始的标准来比较两个游戏。然而，这是低水平的比较，不能给出有意义的解释，如给定游戏实现中可能使用的计算思维模式或者给出转移存在的深刻理解。

上面提到的计算程序行为相似性虽然水平低，但却是语义评价的一种方法。然而，程序员解决问题的方法和编程风格的不同可能导致语义分析不准确。例如，另外两个蜈蚣游戏，看起来相似，玩起来相似，但相似性得分只有0.41。这是因为两个游戏的问题解决方法和编程风格(见表1)存在差异，从而导致不精确的语义游戏分析。这两种蜈蚣游戏规则数量几乎相同，但是它们的代理规则以不同的方式编写。它们都有一个“蘑菇”代理人，即蜈蚣的部分，当被激光击中就变成蘑菇。蜈蚣A使用2个规则实现“蘑菇”代理，而蜈蚣B使用6个规则(见表2、表3)。实现不同导致这两种游戏的相似性得分较低，尽管这两种游戏规则数量几乎相同，玩法也类似。

表1　蜈蚣A和B的结构

	蜈蚣A	蜈蚣B
代理类的数目	8	17
描述的数目	12	34
方法的数目	25	37
规则的数目	109	128

表2　蜈蚣A中蘑菇代理的规则和条件

条件	结果
如果线条上方或下方有3个蘑菇	向下图标移动，向上图标抹去，停止图标抹去
如果线条上方或下方没有蘑菇	设置蘑菇＝蘑菇＋1

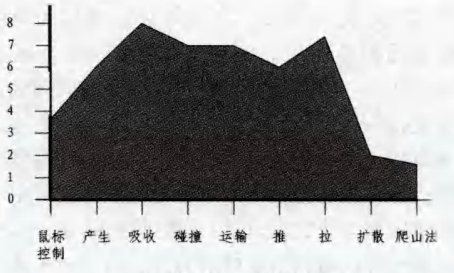
表3　蜈蚣B中蘑菇代理的规则和条件

	条件	结果
如果慢慢分解蘑菇	如果看见“整个蘑菇”图标	变成“大半个蘑菇”图标
	如果看见“大半个蘑菇”图标	变成“蘑菇头”图标
	如果看见“蘑菇头”图标	变成“半个蘑菇头”图标
	如果看见“半个蘑菇头”图标	变成“停止”图标
如果玩家费尽全力修复破坏掉的蘑菇	如果看见“整个蘑菇”图标	什么也不做
	如果什么也没有	变成“整个蘑菇”图标

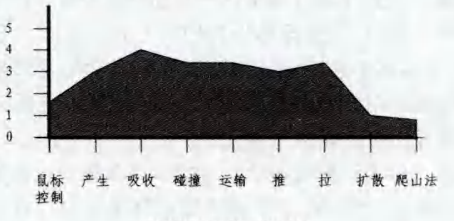
由于相似性得分低，这些游戏看起来不相关，这可能会产生误导。因此，一个高级的语义评价方法，可以检测一个给定的游戏包含的特定计算思维模式，这将是更可取的。为了填补这一空白，本文尝试了以更高级的方法开发计算思维模式图(CTP)。

2.2　计算思维模式图

计算思维模式图是指给定游戏实现中计算思维模式的数量和种类。图2描绘了两种CTP图，确定了9种最流行的计算思维模式，提供具体的语义游戏信息，这些信息通过句法方法无法得到。这9个计算思维模式是已开发多年的游戏集合和科学模拟调查的结果。计算思维模式按实现难度沿横轴方向从左到右排列。为了比较CTP图，在任何给定的CTP图中，计算思维模式的位置以相同顺序保持。CTP图的内部原理是程序行为相似性得分的延伸。CTP图是通过计算给定的AgentSheets项目和9个具有代表性的典型模式之间的程序行为相似性得分绘制的。每个典型模式代表一种计算思维模式，例如“鼠标控制”等。可扩展游戏设计街机(SGDA)可发现这些典型模式。



(a)蜈蚣A的CTP图



(b)蜈蚣B的CTP图

图2　蜈蚣A和B的CTP图

在CTP图中，每个向量得分乘以10，以描述一个给定游戏和每个典型模式(计算思维模式)之间的程序行为相似性得分。同样，每个向量得分代表一个给定游戏中使用了多少计算思维模式。所以，对于一个游戏的特征，如果CTP图形结构中没有，CTP图不会分析这个特征。因此，剩余的计算思维模式是该图的一小部分，未发现的特征会降低那些计算思

维模式的程序行为相似性得分。结果，该游戏的 CTP 图将小于那些只采用 CTP 图形结构中的计算思维模式的游戏。图 2 中上下两个图看起来大小相同，但是它们的缩放比例不同。因为上面的图像最大计算思维模式值是 8，而下面的图像最大计算思维模式值是 4。这种缩放比例差异在图 3 中更明显，即两个 CTP 图重叠了。使用 CTP 图比较上述的两种蜈蚣游戏，图 2 和图 3 显示了更精确的分析。尽管这两种游戏可能实现不同，但它们使用的计算思维模式相同，因为它们是相同的游戏。因此，CTP 图给出了这些游戏的潜在语义的真实情况。

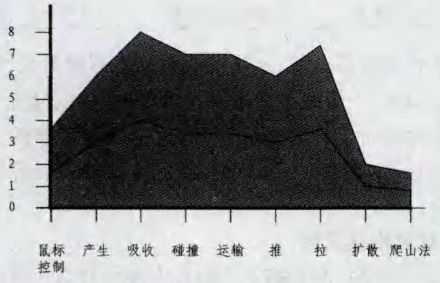


图 3　蜈蚣 A 和 B 的综合 CTP 图

CTP 图能帮助用户如教师或学生更有效地解释和评价游戏。此外，当学生提交自己的游戏到 SGDA 时，CTP 图会自动生成，SGDA 给出即时反馈。SGDA 的作者曾使用系统收集了大约 2500 个 AgentSheets 项目，包括青蛙、推箱子、蜈蚣等街机游戏以及各种来自可扩展游戏设计项目参与者的科学模拟。学生在向 SGDA 提交自己的项目后可以通过 CTP 图获得即时语义评价反馈，并且学生有能力把自己的游戏和 SGDA 中的其他 AgentSheets 项目进行比较。

3　转移

汪应洛等认为，人类学习生活中的必要组成部分中最常用的方法是知识转移[13]。转移是指将某一场合下学到的知识扩展或应用到一个新场合下以解决一个新问题的能力。根据这个定义，所有的学习都可以被认为是一种转移。知识转移可以通过使用多种场合（设置变化越多越好）辅助展示给学生新概念。新知识可以被学生以更抽象的形式保留。当未来新类型的情况出现时，学生可以获取该知识。没有帮助，学生通常不能将纯粹的概念信息转移到真实的现实世界。把任何概念和单一设置或场合联系在一起也会给转移知识到新场合带来困难。所以，尽管转移是我们学习和保留新信息的首选方式，但是转移不能假想任何特定场合。学生先前建立的知识可能加强或阻止吸收新信息。因而，检测可能的知识转移能力有利于多学科领域的研究者。

来自一般的计算机科学领域的学习和知识可能被综合利用到许多其他学科，所以促进计算机科学知识转移到这些领域可以大大提高计算机科学领域内的学习和研究。使用一种工具，可以检测计算机科学以及其他学科知识的潜在转移，往往会增加计算机科学研究的广度和有效性。CTP 图可能表现出知识转移的存在，不仅限于计算机科学相关领域，可用于多学科之间。

CTP 图最初是用来给学生上传他们的游戏到 SGDA 提供反馈。SGDA 为使用 AgentSheets 进行入门游戏编程课程提供提交格式。本学期中，学生接触到简单的计算思维模式；随着课程进展，介绍给他们的将是更复杂和多样的计算思维模式。课程结束时，学生有开放式作业。这些作业鼓励学生根据课堂中获得的最初知识来创建自己的游戏。对于最终设计，学生可选择创建模拟描绘一些自然现象。从语义上分析某个学生的游戏，比较学期开始的和最终设计的（特别是科学模拟），可能发现潜在的知识转移。

例如，图 6 为一个学生创建的混沌理论模拟的 CTP 图，显示了该生如何混合自己先前编写推箱子游戏（见图 4）和 Sims 游戏（见图 5）中学习和使用的计算思维模式。他的科学模拟 CTP 图与推箱子和 Sims 结合的 CTP 图（见图 7）非常相似。因此，对于这个学生，CTP 图显示发生了知识转移。

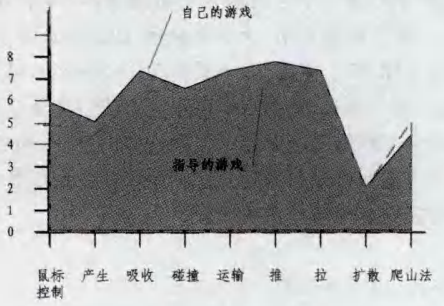


图 4　推箱子游戏的 CTP 图

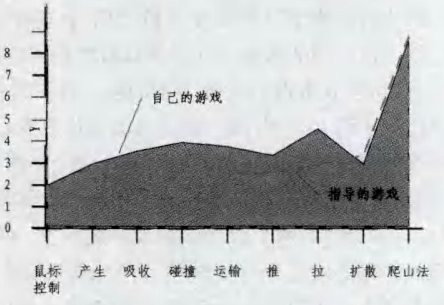


图 5　Sims 游戏的 CTP 图

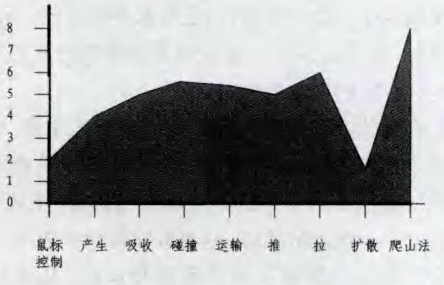


图 6　混沌理论模拟的 CTP 图

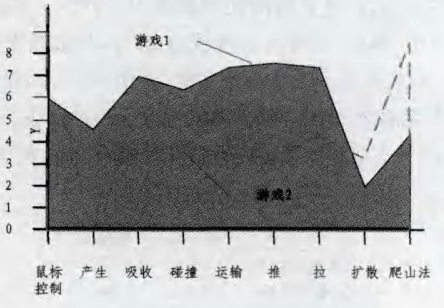


图 7　推箱子-Sims 结合的 CTP 图

结束语 在过去的十年中，可视化编程语言为学生学习编程概念和技能提供了简单的方法，成功地激励了学生。然而，可视化语言并未集中于研究学生创建这些游戏学到了什么类型的知识。计算思维模式图为评估学生获取的具体知识提供初始方法。据观察，对于使用可视化语言进行教育的学校教师和学生，发现教师思维模式的能力是非常重要的。计算思维模式图为我们要回答的问题即“现在学生编程实现俄罗斯方块，学生可以编写科学模拟程序么?”提供了初始方法。此外，计算思维模式图有能力使以人为中心的计算成为可能，如同教师可以获得学生进步的即时反馈。

计算思维模式图的局限性包括指定的计算思维模式的任意性、区分类似模式的难度、计算思维模式图中选择计算思维模式的数量。已选的计算思维模式中，某些如扩散，描述得没有其它准确，如爬山法。尽管这种异常需要进一步研究，但是它没有破坏计算思维模式图的相对精确度，也没有减少检测这些情况下知识转移存在的价值。

分析多种组合的计算思维模式更接近于显示学生知识的深度和广度。计算思维模式图的语义性可让我们评估和可视化一个程序的实际基本意义。一个学生学习的句法评价只显示学生非常有限的背景知识。此外，学生将先前学到的计算思维模式应用到一个科学背景，更清晰地描绘了学生如何将新知识转移到新情况，通过比较 CTP 图可以证明知识转移的存在。在大多数的学习情境中，知识转移常常是假设的，不能保证这些转移实际发生。计算思维模式图是评价知识转移的更好的工具，因为计算思维模式图代表计算思维模式组合为一个可观察的可定义的结果。在一个学期的时间内，通过计算思维模式图检测知识转移的能力，是有效的第一步测量其它领域的转移以及可能的其它形式的学习。

进一步将研究计算思维模式识别的附加验证。目前的模型已经过手动验证，通过比较计算思维模式图的输出和根据玩游戏/模拟以及查看源代码来人工分级评价计算思维模式。计算思维模式图表现相当好，其潜在的误报问题可以通过深化分析水平降低。目前的分析水平停留在个别条件和行为上。分析没有将这些行为和条件分解成参数，这些参数可以更有效地区分类似模式。

参 考 文 献

[1] Repenning A, Webb D, Ioannidou A. Scalable game design and the development of a checklist for getting computational thinking into public schools[C]// Proc. SIGCSE'10. ACM Press, WI, USA, 2010

[2] 周以真. 计算思维[J]. 中国计算机学会通讯, 2007, 3(11)

[3] Wing J M. Computational Thinking[J]. Communications of the ACM, 2006, 49(3)

[4] 董荣胜，古天龙. 计算思维与计算机方法论[J]. 计算机科学, 2009, 36(1)

[5] Kozaczynski V, Ning J, Engberts A. Program Concept Recognition and Transformation[J]. IEEE Trans. On Software Eengineering, 1992, 18(12): 1065-1075

[6] Duscasse S, Rieger M, Demeyer S. A Language Independent Approach for Detecting Duplicated Code[C]// Int'l Conf. on Software Maintenance. 1999: 109-118

[7] 苏舟. 基于向量空间范围搜索的大型软件相似度检测[D]. 杭州：浙江大学, 2008

[8] 李亚军，徐宝文，周晓宇. 基于 AST 的克隆序列与克隆类识别[J]. 东南大学学报, 2008, 38(2): 228-232

[9] Krinke J. Identifying Similar Code with Program Dependence Graphs[C]// Proceedings Eighth Working Conference on Reverse Engineering. 2001: 301-309

[10] Lewis C M. How programming environment shapes perceptionlearning and goals: Logo vs. Scratch[C]// Proc. SIGCSE'10. ACM Press, WI, USA, 2010

[11] 朱国强，刘真，李宗伯. 对计算机系统中程序行为的分析和研究[J]. 计算机应用, 2005(12)

[12] 陈浩，王广南，孙建华. 一种基于图的程序行为相似性比较方法[J]. 计算机应用研究, 2010(2)

[13] 汪应洛，李勖. 知识的转移特性研究[J]. 系统工程理论与实践, 2002(10)

(上接第 370 页)

从检索结果可以看出，在对指定资料库进行文件名(null)和内容项(Lucene)关键词搜索时，一共找出 7 个相关文件，包括 WORD、PDF、TXT 格式文档。在“查询结果”输出框中，给出了相关文件内容关键词的上下文信息，信息断句自然合理。整个检索用时 31ms，效率较高，达到了预期研发目的。

结束语 本文阐述了当前计算机本地资源搜索存在的问题，介绍了全文检索引擎 Lucene 的基本原理、源码和功能结构，实现了一个利用 Lucene 来解决本地搜索问题的搜索引擎。引擎有效地运用了 Lucene 全文检索的特性，通过将 WORD、PDF 等常见文档解析构造成 Document 对象并对文档内容进行分片索引，实现了对文档内容的全文检索。下一步工作将对引擎的动态索引进行研究，定期检测路径下文件的变动信息，以保证检索的时效性。

参 考 文 献

[1] Gospodnetic O, Hatcher E. Lucene in action[M]. Manning Publications Co., 2005

[2] 孙西全，马瑞芳，李燕灵. 基于 Lucene 的信息检索的研究与应用[J]. 情报理论与实践, 2006, 29(1): 521-528

[3] 邱哲，符滔滔，王学松. 开发自己的搜索引擎 Lucene+Heritrix (第 2 版)[M]. 人民邮电出版社, 2005

[4] 林洁. 个性化综合倒排索引在 Lucene 中的应用[J]. 电脑知识与技术, 2010, 6(4): 932-934

[5] PDF-百度百科[OL]. http://wapbaike.baidu.com/view/15963.htm

[6] 包宇宁. 使用 Java 编程解析 HTML 文档[J]. 福建电脑, 2004(9): 86-87

[7] 周筱媛. 用 Java 集合类处理 XML 文档[J]. 西安科技学院学报, 2002, 22(3): 318-320