★CACM 中国版★

计算机协会通讯

CACM.ACM.ORG

2015 年 8 月第 58 卷第 8 期

量子编程的未来

计算创造性

测试 Web 应用程序

网络、万维网和互联网科学

Soylent

Association for Computing Machinery

ACM 通讯

精选文章中文译文

ACM中国理事会办公室

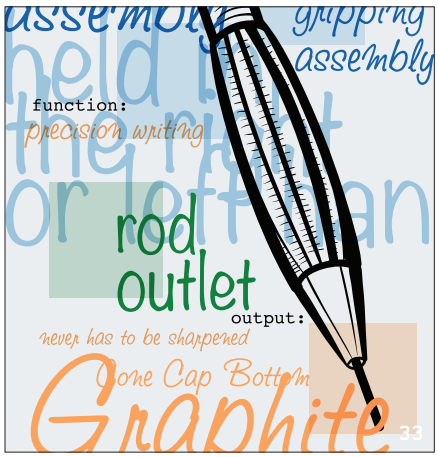
中国北京清华大学
东主楼 11-236 室
邮编：100084
电话：+86-10-62785025
电子邮件：acmchina@acm.org
联系人：辛爽

ACM通讯

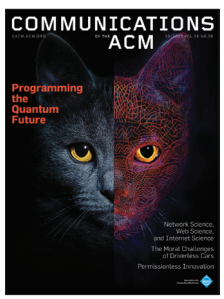
(ISSN 0001-0782) 由计算机协会（2 Penn Plaza, Suite 701, New York, NY 10121-0701）按月发行。

Association for Computing Machinery

观点

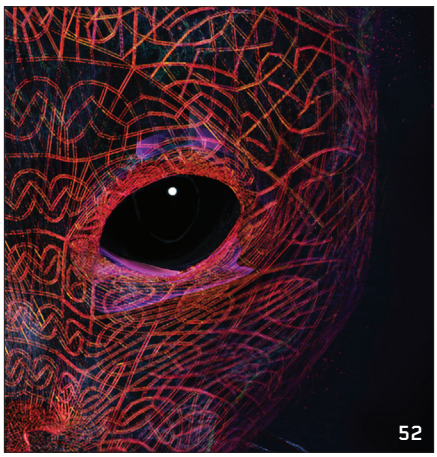


实践



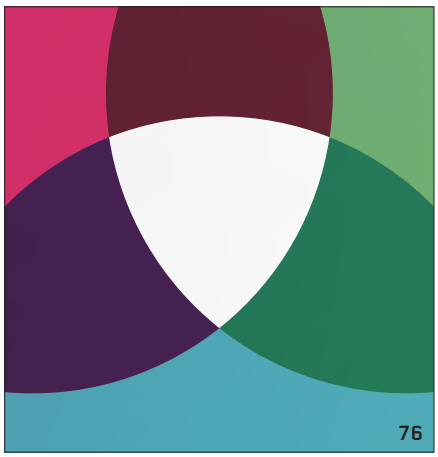
关于封面：
几十年来，薛定谔的猫这一标志性图像一直用来说明量子力学中新兴理论之间的差异，在本月的封面故事中来21世纪走了走（第52页）。此文探讨了量子编程语言，并提出一个量子计算模型。封面插图由 Peter Bollinger 提供

投稿文章



观看此独家《通讯》视频中作者对本研究的讨论：http://cacm.acm.org/videos/programming-the-quantum-future

评论文章



研究亮点

在本《通讯》独家视频http://cacm.acm.org/videos/soylent-a-word-processor-with-a-crowd-inside 中，您可以观看作者对本研究的讨论。

Association for Computing Machinery
Advancing Computing as a Science & Profession

图像从左至右由：ANDRIJ BORYS ASSOCIATES/SHUTTERSTOCK，FUTUREDELUXE，IWONA USAKIEWICZ/ANDRIJ BORYS ASSOCIATES 提供

DOI:10.1145/2699391 Leen-Kiat Soh、Duane F. Shell、Elizabeth Ingraham、Stephen Ramsay、Brian Moore

观点
通过计算创造性来学习

通过把创造性思维纳入课程体系，
可以增进学生在计算机科学入门课程中的学习并提高学习成果。

美国国家科学基金会的“重建多样性（Rebuilding the Mosaic）”报告中[a]提到，在处理所有领域中出现的新问题时，均需要使用和管理大型数据库，创造性地设计以数据为中心的问题解决方案，以及应用计算和计算机方面的思维方法进行跨学科研究。为了应对这些需求，计算机科学（CS）入门课程的作用逐步超出了一门普通的 CS 专业课程。这门课程越来越大的作用包括：它的课程设计不仅需要考虑为未来的 CS 科学家和从业人员打基础，还要激发学生对 CS 的兴趣、动员和吸引新的学生加入 CS；此外，它还要为其他专业的学生提供计算思维的方法和 CS 方面的技能；它甚至可以为未来的 CS K-12 教员提供培训。CS 入门课程的这种多面性要求人们采用新的方式设计 CS 课程体系。除了计算思维之外，人们还把创造力单独拿出来作为解决重要社会问题的关键因素和二十一世纪的核心技能（比如，2012 年国家研究委员会报告）。受上述现象的影响，我们看到了一个把创造性思维纳入课程体系来修订计算机科学入门课程的良机。

a 请参阅http://www.nsf.gov/pubs/2011/nsf11086/nsf11086.pdf.

创造性思维

创造性思维并不是与生俱来的天赋，也不是只有少数人能驰骋的疆域，它更不是仅仅局限于艺术领域。与此相反，它是人类智力发展的内在过程，可在任何场景下练习、激励和开发。[1,2,5,7–9]Epstein 的生成理论（Generativity Theory）[2] 认为创造性思维由下列四个核心能力组成：

- *拓宽知识面（Broadening）*。人的知识和技能面越宽，能创造出的各种可行的更多变新奇模式和组合的范围也就越多，也更吸引人。因此为了获得创造力，人们必须获取除当前的学习和专长领域之外的信息和技能，拓宽自己的知识面。
- *挑战（Challenging）*。创新总是源于当前不够有效的策略和方式。挑战的难度越大，越有可能激发出创新的解决方案。
- *周边环境（Surrounding）*。多样、混乱的情况和刺激因素会创造出能够让新奇的战略和行为萌发的环境——例如，从新的角度看待事物，与新的人群打交道以及考虑多种感官表象。
- *捕捉（Capturing）*。新的想法会一直出现，但是大多数时候人们都没有注意到。创造性要求人们留心和记录出现的新想法。

在当代的认知和神经科学研究中，这些核心能力有着稳固的根基。[6] 正如 Wing[10,11] 提出了令人信服的实例来论证计算思维的广泛性，我们也可以论证，Epstein 的核心创造性思维能力也是一种可广泛应用的技能集，这个技能集不仅为创造性地应用现有知识和技能提供了准

备，而且还为人们参与终生学习、增强他们在跨学科背景下处理日益复杂的问题的能力奠定了基础。

计算创造性方面的练习

在我们的训练框架中，计算思维和创造性思维两者均被视为认知手段，这两者的结合会产生出新的*计算创造性*。但是不能把这种结合当成一种二分的结合，它是一种共生的能力和方法。计算思维和 CS 技能扩展了人们能够使用的知识和方法，从而*拓宽了*问题解决方案的范围。*挑战性的*问题迫使人们用前所未有的不寻常的方式使用计算手段，从而为新老问题引出了新的计算方法。当*周边*环境是新的环境和合作者时，有助于我们产生看待问题的新方式，并让我们注意到不同的刺激因素或者从计算的角度思考问题最后，通过*捕捉*数据表示和算法方面的想法，把人们引向新的数据结构和求解过程。通过融合计算和创造性思维，学生们可以利用他们的创造性思维能力来“开启”他们对计算思维的理解。[6]

我们创建了用于提高学生的计算创造性计算创造力练习（CCE）套件。每个 CCE 有四个共同的部分：目标、任务、CS 灯泡——解释与 CS 有关的概念、思路和实践的载体——以及直接把练习任务与 CS 主题相关联的学习目标。我们的计算创造性练习的设计基于以下原则：平衡计算思维和创造性思维之间的各种因素以及在计算和创造性方面的概念和技能之间进行映射，后者是出于它们分别表现在不同领域中的考虑。每个 CCE 要求每位学生花上约一到两个小时，但因为练习需要协作，所以学生们会有两周的时间进行练习。在 CCE 的设计中，第 1 周首先给学生安排了动手实践和团队任务，在第 2 周则通过解答分析和回顾问题来反思第 1 周的活动。第 1 周和第 2 周的表现都会评分。

举例而言，在*日常对象 CCE*（*Everyday Object CCE*）中，我们要求学生发明我们可能经常用到的普通物品。其中的挑战在于想象这件物品并不存在，然后用书面语言描绘下列因素：所选物品的机械功能；该物品所满足的需要；以及它的物理属性和特点。这种描述必须足够具体，具体到能让从未见过该物体的人认出它，理解它的工作原理并了解它所提供的好处。（注：我们给学生列了一个物品清单以供选择，其中包括拉链、自动铅笔、装订夹、Ziploc 袋子、剪刀、卷尺、订书机、指甲钳、雨伞、手电筒、开罐器、衣夹、便利贴、厕纸架、旋转门。）然后，我们要求学生考虑下列问题，并写出他们的回答。

分析：（1）把您的对象当成一个计算机程序。画图，在图中用方框表示它的所有功能（并为之命名），然后标出每个功能的输入和输出。在这些功能中，有没有共同的输入和输出？（2）列一个由物理属性和特性组成的列表。组织这些属性和特性，把每一个属性和特性声明为具有合适类型的变量。某些属性 / 特性是否可以被排列成一种包含相关属性 / 特性的层级结构？***反思：***（1）考虑您对分析 1 的回答，是否可以整合一些功能，从而更简洁的来表示对象？是否应该引入一些新功能，以便更好地描述您的对象，从而让这些功能更具模块化？（2）您是否听说过抽象？计算机科学中的抽象与您在本练习中完成的功能和特性识别过程是否有什么关联？

我们基于教学设计原则的 CCE 对深度学习，举一反三和发展交际能力都有所影响。其设计可以让人们通过结合亲自动手、基于问题的学习与书面分析和反思来进行 CS 概念的教学。通过从更抽象的角度使用计算思维和 CS 内容，CCE 更好的实现了这种知识迁移，同时也不需要使用程序代码来处理似乎与 CS 无关的问题。CCE 还促进了创造性能力的发展，这个实现是通过使用多种感官渠道，要求学生使用整体的、富有想象力的思维方式，向他们提出了具有挑战性的问题并利用个人和协作式的团队工作来帮助他们发展人际交往技巧。CCE 利用了认知神经科学、认知科学和心理学方面的综合研究，涉及统一学习模型（Unified Learning Mod-

我们看到了一个把创造性思维纳入课程体系来修订计算机科学入门课程的良机。

el）[6] 中确定的注意（*attention*）、重复（*repetition*）和联系（*connection*）等认知 / 神经学习过程。通过专注于计算思维的原理，增强了课程材料的学习效果和记忆程度，在课堂中提供了更多的计算思考和计算概念的重复机会，并在更高层级的抽象层面把材料与更丰富的场景和应用联系起来。

一些证据

2012 年秋季学期期间，我们在内布拉斯加大学林肯分校的四门不同的计算机科学入门课程中设置了 CCE。每门课程均针对不同的目标群体（CS 专业、工程专业、混合 CS/ 物理科学的专业以及人文学科的专业）进行了调整。来自 150 名学生的研究结果说明，从控制组的累计 GPA 方面来说，完成的 CCE 的数量与课程的分数显著相关（$F(3, 109) = 4.32$，$p = .006$，partial $Eta^2 = .106$）。在完成 2 到 4 项练习的区间内，存在显著的线性趋势（$p = .0001$）。完成的 CCE 的数量还与计算思维知识的测试分数显著相关（$F(3, 98) = 4.76$, $p = .004$, partial $Eta^2 = .127$）。 类似的，在完成 0–1 到 4 项练习的区间内，存在显著的线性趋势 ($p < .0001$)，且 CS 和非 CS 专业没有区别。[3,4] 这些结果说明了一种“剂量（dosage）” 效应，多完成一项 CCE，课程分数和测试分数就会增加。这种增加不容轻视。因为就每一项完成的 CCE 而言，在知识测试中，学生的成绩提高了约一个等级以及约一个绩点。

在第二次评估中，[7] 我们使用了准实验设计，为对比 CCE 的实施情况，在 2013 春季学期安排针对工程师的计算机科学入门课程（$N = 90$，96% 的人完成了三项或四项练习），控制组 / 对照组是未安排 CCE 课程的 2013 年秋季学期（$N = 65$）。使用协方差分析（ANCOVA）后，我们发现，对于在安排 CCE 的学期中学习的学生，他们的计算思维知识测试成绩明显高于在受控学期中学习的学生成绩（$M = 7.47$ 到 $M = 5.94$），控制的各个因素包括学生的课程成绩、战略性的自我调节、参与程度、动力和课堂感知。（$F(1, 106) = 12.78$, $p < .01$, partial $Eta^2 = .108$）。与未参与 CCE 的学生相比，在把计算机科学知识和技能应用到工程方面，参与 CCE 的学生还展现出了更高的自我效能感（$M = 70.64$ to $M = 61.47$; $F(1, 153) = 12.22$, $p < .01$, partial $Eta^2 = .074$）。

在当代的认知和神经科学研究中，这些核心能力有着稳固的根基。

总体来说，与课堂教育干预的传统发现结果不同是，这些新发现有很强的、具有现实意义的影响。对于 CS 专业和非 CS 专业而言，这些练习似乎都积极地影响了核心课程内容的学习过程和学习成果。这些发现验证了我们的，即*计算创造性的练习能够把 CS 中的计算概念带给 CS 和类似的非 CS 学科的学生，并增强他们对计算思维的理解*。

行动的召唤

受到上述评估结果的鼓舞，现在我们正着手改编 CCE 使之适应中等教育，根据 CCE 的套件设计一门课程以及继续开发更多的 CCE。不仅如此，把计算创造性纳入 CS 入门课程还具有更广的影响，其范围包括触及 CS 中的弱势群体，让年轻的学习者接触计算创造性，改善 CS 的学习情况以及为学生打好基础，帮助他们在与日俱增的跨学科领域中成为创造性的问题解决者。因此，我们呼吁 CS（和其他 STEM）的教育家行动起来，在他们的课程或课程体系中调研和实施计算创造性。 C

参考资料

1. Epstein, R. *Cognition, Creativity, and Behavior:Selected Essays*.Praeger, 1996.
2. Epstein, R. Generativity theory and creativity.*Theories of Creativity*.Hampton Press, 2005.
3. Miller, L.D. et al.Improving learning of computational thinking using creative thinking exercises in CS-1 computer science courses.*FIE* 43, (2013), 1426–1432.
4. Miller, L.D. et al.Integrating computational and creative thinking to improve learning and performance in CS1. *SIGCSE'2014* (2014), 475-480.
5. Robinson, K. *Out of Our Minds:Learning to be Creative*. Capstone, 2001.
6. Shell, D.F., Brooks, D.W., Trainin, G., Wilson, K., Kauffman, D.F., and Herr, L. *The Unified Learning Model:How Motivational, Cognitive, and Neurobiological Sciences Inform Best Teaching Practices*.Springer, 2010.
7. Shell, D.F. et al.Improving learning of computational thinking using computational creativity exercises in a college CS1 computer science course for engineers. *FIE* 44, to appear.
8. Shell, D.F. and Soh, L.-K.Profiles of motivated self-regulation in college computer science courses:Differences in major versus required non-major courses.*J. Sci.Edu.Tech.Technology* (2013).
9. Tharp, T. *The Creative Habit:Learn it and Use it for Life*. Simon & Schuster, 2005.
10. Wing, J. Computational thinking.*Commun.ACM 49*, 3 (Mar. 2006), 33–35.
11. Wing, J. Computational thinking:What and why.*Link Magazine* (2010).

Leen-Kiat Soh （lksoh@cse.unl.edu）是内布拉斯加大学计算机科学和工程系副教授。

Duane F. Shell （dshell2@unl.edu）是内布拉斯加大学研究教授。

Elizabeth Ingraham （eingraham2@unl.edu）是内布拉斯加大学文学副教授。

Stephen Ramsay （sramsay.unl@gmail.com）是内布拉斯加大学英语Susan J. Rosowski副教授。

Brian Moore （brian.moore@unl.edu）是内布拉斯加大学音乐教育与音乐技术副教授。

本材料基于美国国家科学基金会资助的研究（资助号1122956）。其他资助由内布拉斯加大学林肯分校（UNL）Phase II Pathways to Interdisciplinary Research Centers （至跨学科研究中心第二阶段通道）提供。本文中的所有观点、发现、结论或建议仅属于作者本人，并不一定反映美国国家科学基金会或UNL的观点。

译文责任编辑：李向阳

DOI:10.1145/2755501

文章编写主导者 acmqueue
queue.acm.org

状态驱动测试。

作者：ARIE VAN DEURSEN

利用状态对象对 Web 应用程序进行测试

对 WEB 应用程序进行端到端测试时，往往需要通过像 Selenium WebDriver 这样的框架与网页进行复杂的交互。[12] 要将网页的这种复杂性隐藏起来，推荐的方法是使用页面对象，[10] 但首先要解决几个问题：测试 Web 应用程序时，应该创建哪些页面对象？应该在页面对象中包含哪些动作？根据页面对象，应该指定哪些测试场景？

过去的几个月里，我一直在使用页面对象测试一款 AngularJS（https://angularjs.org）Web 应用程序，在此期间我找到了以上问题的答案，方法是将页面对象抽象到状态层。将 Web 应用程序视为状态图，极大地简化了测试场景和对应的页面对象的设计过程。本文描述了这个过程逐渐成形的方法：本质上，根据状态生成页面对象，后文简称为*状态对象*。

WebDriver 是一款融合了最先进技术的工具，广泛应用于测试 Web 应用程序。它提供了 API，可以访问在浏览器中渲染的网页元素。例如，它可以通过访问表格包含的文本，或元素的样式属性来检查元素。另外，该 API 还可以与页面交互——例如，点击链接或按钮，或在输入表单中输入文本。因此，WebDriver 可以用来通过 Web 应用程序的点击场景编写测试脚本，形成应用程序的端到端测试套件。

WebDriver 支持用你选择的任何一款浏览器测试（例如 Internet Explorer、Firefox、Chrome 等）。它还支持多种语言绑定，因此你可以用 C#、Java、JavaScript 或 Python 编写场景。

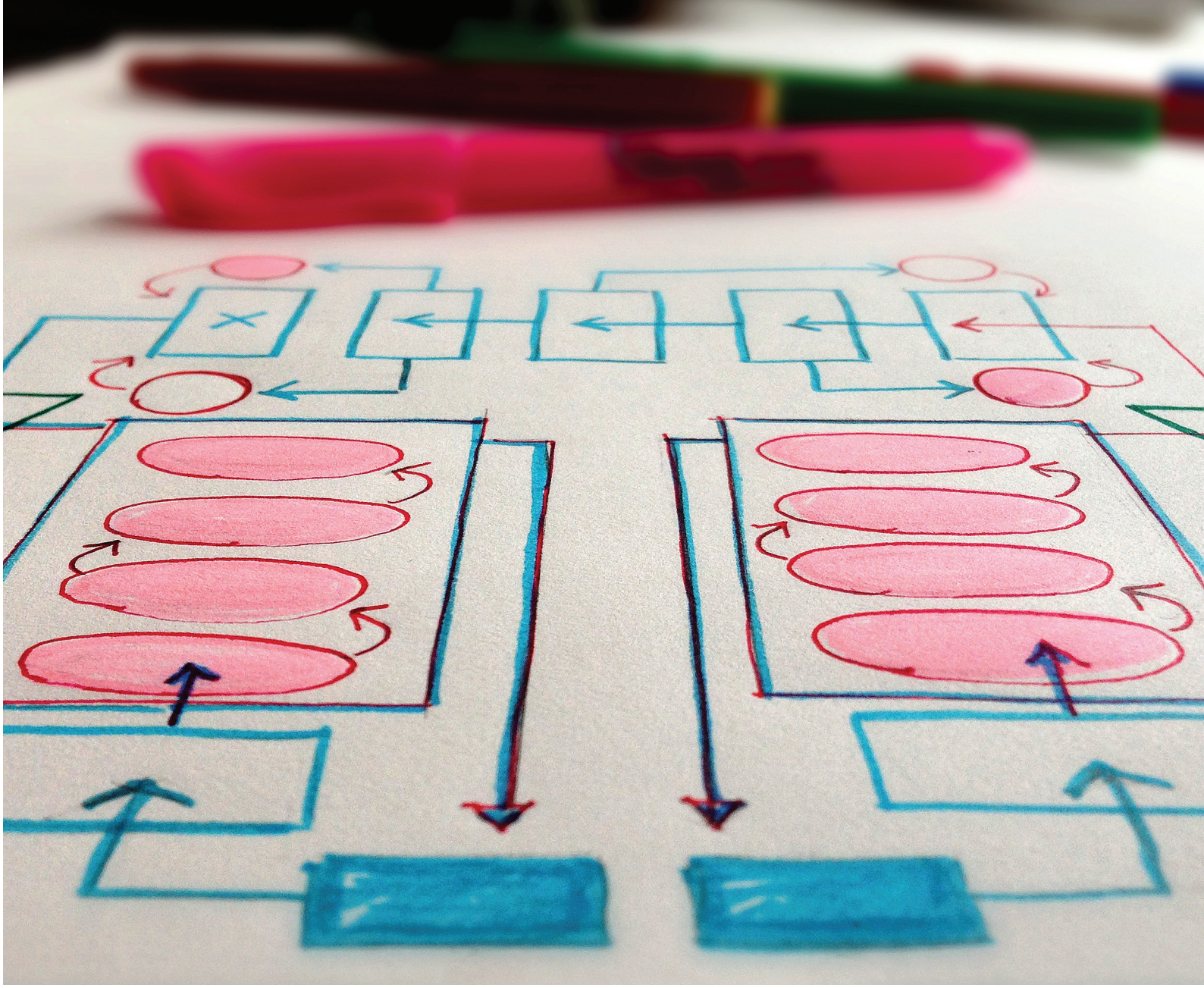
页面对象。WebDriver 提供了 API，可以访问在浏览器中渲染的网页元素。然而，要让终端用户场景有意义，就不应该用网页元素来表达，而是用*应用程序域*。

因此，建议用页面对象编写 WebDriver 测试。此类对象会提供 API，在网页元素之上实现领域概念，如图 1 所示，此图截取自软件设计师马丁·福勒 (Martin Fowler) 制作的一副插图。[4]

这些页面对象会隐藏定位器所使用的特定元素（例如用于查找按钮或链接的元素）和下层“插件设计”的细节。这样一来，测试场景的可读性提高了，当页面详细信息发生变化时也更容易维护。

页面对象不需要表示整个页面；它们也可以表示用户界面的一部分，例如导航面板或上传按钮。

要表示应用程序中的导航，“页面对象上的方法调用应该返回其他页面对象。”[10] 本文正是基于这个想法，更进一步，提出了所说的*状态对象*。



照片由 ALICIA KUBISTA/ANDRIJ BORYS ASSOCIATES 提供

用状态图为 Web 应用建模

我们使用基于统一建模语言 (UML) 的状态图来建立 Web 应用程序的导航模型。图 2 展示的是登录某个应用程序的状态图。用户或处于身份认证的过程中，或已完成身份认证。他们最初未通过身份认证，然后输入凭证，如果凭证正确，他们就到达身份已认证的状态。身份认证后，用户可以注销并回到进行身份认证的页面。

该状态图通常会指向两个页面对象：

- 一个对象是登录页面，对应“正在认证身份”状态。
- 另一个对象是注销按钮，显示在处于“身份已认证”状态的所有页面上。

为了强调这些表示状态的页面对象，它们被赋予了明确的功能用于状态导航和状态检测，从而成为了状态对象。

状态对象：检测并触发方法

每个状态对象都包含两种方法：

- *检测方法*会返回浏览器显示的处于给定状态的关键元素的值，例如用户名、文件名或某些指标的值；我们可以在测试场景中利用这些方法验证浏览器是否显示了预期的值。
- *触发方法*对应一次模拟的用户点击，并触发浏览器转移到新状态。处于“正在认证身份”状态时，用户可以输入凭证并点击提交按钮，假设凭证正确，浏览器将进入下一个“身份已认证”状态。之后，用户可以点击注销按钮，回到“正在认证身份”状态。

一个有用的方法是，将最重要的检测方法组装成属性的自校验程序，每当程序处于特定的状态，这些属性都必需满足。例如，处于“正在验证身份”状态时，应该会有用于输入用户名或密码的字段；应该还会有一个提交按钮；或许 URL 中应该包含登录路径。这样的自校验方法可以用于验证浏览器是否确实处于给定状态。

场景：触发和检测事件。给定一组状态对象，测试案例通过状态机描述相关的*场景*（路径）。例如：

1. 前往登录 URL。
2. 通过自校验验证是否处于“正在认证身份”状态。

图 1.页面对象。（插图作者：马丁·福勒 (Martin Fowler)）

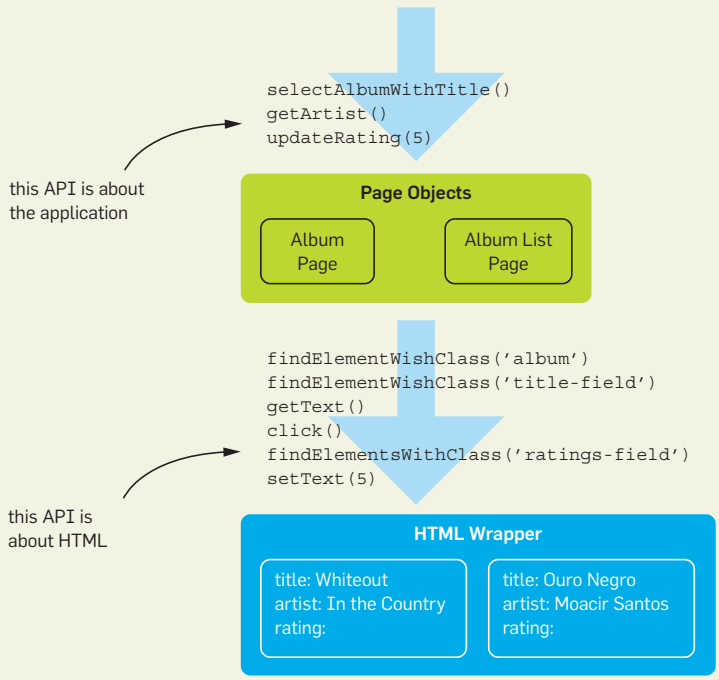
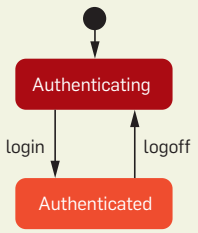


图 2.登录应用程序。



5. 点击关闭。
6. 执行身份认证自校验。

图 3 中的箭头符合如下形式

```
事件 [ 条件 ] / 动作
```

因此，触发器（点击）可以是有条件的，除了引向新状态之外，它也可以触发动作（服务器端）。

测试此类转换时，要先触发事件，确保满足条件，然后验证能否观察到任何所需动作的效果，以及是否会到达对应的状态。

图 3.通过条件事件登录。

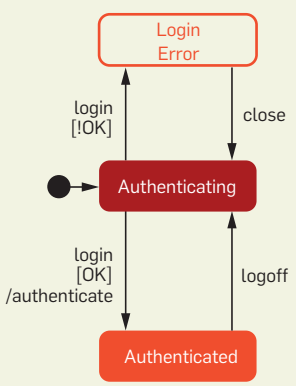
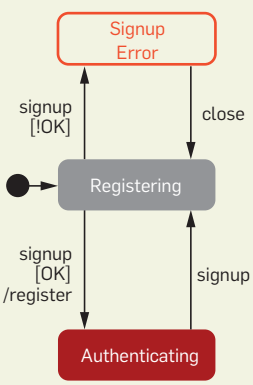


图 4.注册新用户。



3. 输入正确的凭证并提交。
4. 验证是否处于“已认证身份”状态。
5. 点击注销。
6. 验证是否处于“正在认证身份”状态。

这样，一个场景（测试或验收）就是一个动作序列，每个动作结束后都会检测是否到达预期的状态。

有条件的事件。除成功的登录外，真实的登录流程还应该处理使用无效凭证登录的情况，如图 3 所示。该图展示了一种额外的状态，这种状态下会显示一条错误信息。这种额外的状态将产生一个额外的状态对象，对应相关信息的显示。就动作而言，它只有一个关闭按钮引导用户返回原来的登录页面。

该额外状态自然而然地引出了另一个测试场景：

1. 前往登录 URL。
2. 执行身份认证自校验。
3. 输入无效的凭证并提交。
4. 执行登录错误自校验。

扩展状态图

为了驱动测试，可以扩展状态图，以满足其他测试场景的要求。例如，注册新用户与身份认证有关，如图 4 所示。此图包含“正在认证身份”状态，但并不包括它全部的流出箭头。相反，此图的焦点在于一个新的*正在注册*状态以及该状态下可能出现的转换。

这就产生了另外两个新的状态对象（分别对应注册和显示错误信息）以及两个额外场景。因此，当为给定页面设计测试时，不必考虑完整的状态机：关注感兴趣的状态就足以设计出测试案例了。

超状态。具有相同行为的状态可以组合成为*超状态*（又称 OR 状态）。例如，一旦通过身份认证，所有页面可能会有相同的页眉，页眉中含有注销按钮和导航至关键页面的按钮（例如，帐户管理按钮和获取帮助按钮，如图 5 所示）。

从超状态（例如 `注销`）引出的边是四种内部状态（子状态）对于 `注销` 事件响应的简化。该简化版扩

展后的情况如图 6 所示，从超状态引出的 5 条有向边已根据四种内部状态进行扩展，这样就得到了 4 × 5 = 20 条（定向）边（画成双向边是为了方便维护状态图）。

超状态常用可重用的 HTML 片段实现，例如，在 AngularJS 中通过 ngInclude 实现。[1]

在此类情况下，创建与公用的包含文件对应的状态对象是最自然的做法。它包含对所需链接或按钮的存在检测和事件检测，例如用来判断点击设置按钮是否的确会进入设置状态。

因而测试场景可能会是这样的:

1. [登录所需的步骤。]
2. 执行文件夹自校验。
3. 点击设置链接。
4. 执行设置自校验。
5. 点击帮助链接。
6. 执行帮助自校验。
7. 点击帐户链接。
8. 执行帐户自校验。
9. 点击文件夹链接。
10. 执行文件夹自校验。
11. 点击注销链接。
12. 执行身份认证自校验。

这对应于身份认证后导航面板的一个测试场景。它测试的是在帮助页面点击帐户链接是否正确。但是，它不会校验在设置页面点击帐户链接是否正确。实际上，该测试仅包含扩展版状态图的 20 个状态转移中的的四个。

当然，可以为所有的 20 条状态转移边设计测试。如果待测试应用的导航面板是每次查看页面时产生的，而不是通过一个单独的 include 文件产生的，那么这样做就有意义。在这种情况下，我们有理由怀疑不同的路径可能会暴露不同的错误。但是通常来说，如果采用 include 文件设置，就没有必要测试所有的扩展路径。

状态遍历。针对导航页眉的单一测试场景会访问五种不同的状态，并且顺序是不变的，如图 7 所示。这是一个很长的测试，可以拆分为四个单独的测试案例（图 8）。

在单元测试中，推荐采用第二种方案。这个方案的优点是四个测试互相独立：其中一个步骤的失败不会影响其他步骤的测试。而且，故障诊断也更容易，因为失败的测试案例更加清晰。

然而，四个互相独立的测试很可能导致耗时大幅延长：每个测试都要进行完整的身份认证，这样会极大地拖慢测试执行进度。因此，在端到端测试中，不同的测试案例常常共享同样的初始化设置步骤。

就 JUnit（http://junit.org）的设置方法而言，[9] 单元测试套件通常会使用 @Before 设置，该设置会在类中的每个 @Test 之前反复执行。与此相反，端到端测试更可能使用 @BeforeClass，目的是仅执行一次繁复的设置方法。

模式对话通常用来禁用所有交互，直至用户对重要的消息作出确认（例如，“您确定要……？”）。前文展示的登录或注册错误信息就是实例。此类模式对话需要使用一个单独的状态，也就是一个特殊的页面对象，可以通过一个接受事件来结束对话。

模式对话可以用浏览器警告信息（在 WebDriver 中，确认警告信息后，测试才能继续）或 JavaScript 逻辑来实现。对于第二种情况，可能还需要额外测试的是对话是否确为模式对话（换言之，是否其他所有与页面的交互都被禁用了）。

图 5.超状态的共有行为。

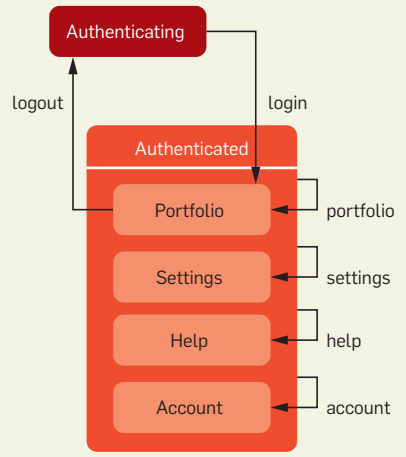


图 6.扩展超状态。

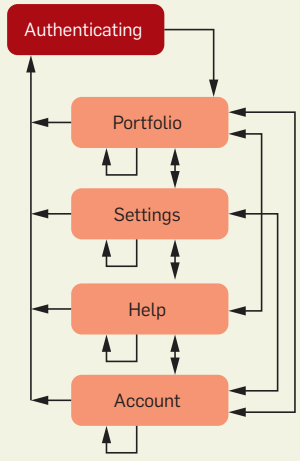


图 7.遍历多个状态的单一测试。

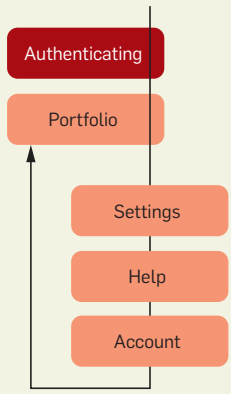


图 8.针对不同状态的单独测试。

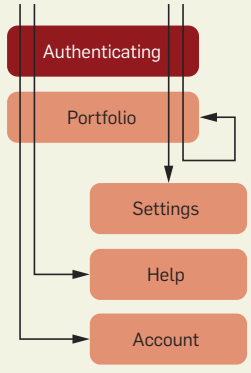


图 9.带有错误处理和共享导航的登录状态机。

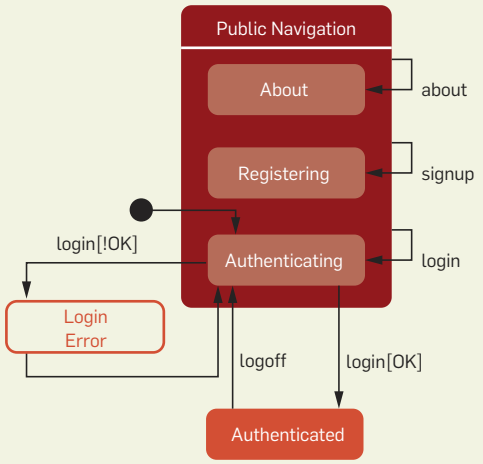


图 10.利用转换注释来简化状态图。

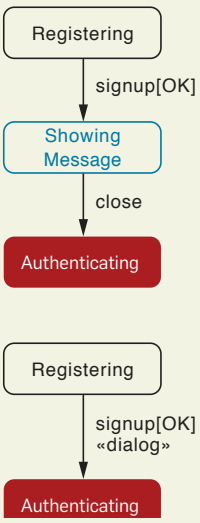
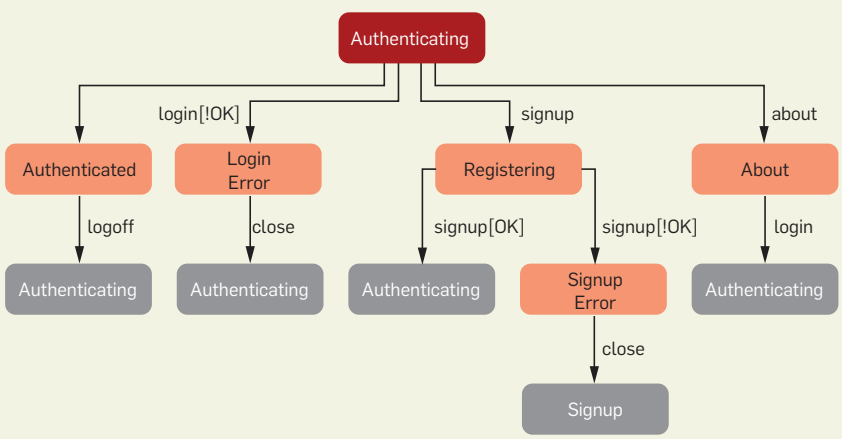


图 11.转换树。



如果模式对话是被超状态所含的状态触发的，那么该对话状态不应该属于该超状态（因为对话中的超状态交互已禁用）。因此，针对错误处理和共享导航的登录过程，绘制状态机的正确方法应如图 9 所示。此处的错误对话不属于导航超状态，因为它仅允许关闭事件，不允许点击，例如不可点击“关于”的链接。

有些应用程序含有大量对话，例如，处理注册、登录、忘记密码等情况的对话。许多这样的对话唯一的用途是通知用户状态已更改成功。为了简化状态图，可将对话作为注释标在边上，如图 10 所示。

上方的图是完整版本，下方的图是简化版本。请注意，那条 <<dialog>> 注释对测试实现很重要。测试案例*必须*点击对话的关闭按钮；否则测试将阻塞。

转换树。为了更好地分析状态的可到达性，以及状态和转换的覆盖情况，可将状态图转化为转换树，如图 11 所示。（如需进一步了解，请参考 Robert Binder 的《*Testing Object-oriented Systems*》[3] 一书中关于测试状态机的章节。）

图 11 显示了前文中注册和身份验证信息的状态机所对应的转换树。从最初的“正在认证身份”状态开始，我们先对图进行宽度优先遍历。因此，从首先访问的状态进入其直接相邻的后续状态。如果你进入已经访问过的状态，该访问过的状态就会显示为灰色的叶节点。然后进入下一个未访问过的状态。

这棵树能帮助我们为单一状态设计测试：树中到达该状态的路径是图中到达该状态的最短路径。此外，它清楚地显示了一个状态有哪些转移的边。

这棵树还能帮助我们为完整状态机设计测试套件：为每条路径写一个从根节点到叶节点的测试案例，得到的就是一个覆盖所有转换的测试套件，因而它也能覆盖状态机的所有状态。

覆盖路径。尽管关注各个单一状态及其转换是定位和排除基本故障的好方法，但一些复杂的缺陷，只有沿一条*路径* 跟踪多个状态时才能发现。

例如，想一想客户端缓存的例子。在像 AngularJS 这样的 Web 开发框架中，可以很容易地启用（部分）后端 HTTP 调用。只要方法得当，就能改善响应速度、减少网络往返通信，因为后端调用的结果被缓存记录下来了，不需要反复发送请求。

但是，如果调用结果容易发生变化，就可能导致不正确的结果出现。例如，客户端可能会请求一个

包含全部所需信息的概述页面，修改另一个页面的下层数据，然后返回最初的概述页面。这对应于图 12 中的红色路径。

启用缓存后，文件夹的状态将缓存后端调用的结果。在发生改变时，如果设置状态下的缓存实现正确，将导致缓存无效。结果就是当再次访问“文件夹”状态时，会再次发出请求，而且会使用更新后的结果。

针对这类缓存行为的测试案例可能会像这样：

1. [通过最短路径到达“文件夹”状态。]
2. 从“文件夹”中搜集感兴趣的值。
3. 点击设置链接，导航至设置。
4. 修改会影响感兴趣的“文件夹”值的设置。
5. 点击“文件夹”链接，导航回到“文件夹”。
6. 确认正确显示修改值。

在前文提到的 AngularJS 应用程序中，此测试案例捕捉到了一个真实存在的错误。不幸的是，要设计出一种的测试策略，覆盖*所有*可能含有错误的路径，是非常困难的或成本很高。

通常来说，当有循环时，要跟踪的潜在路径有无数个。因此，测试人员需要凭借专业知识找出感兴趣的路径。

前文所述的基于转换树的方法提供了所谓的往返覆盖 [2]，即在回到已经访问过的节点之前，它会将每个循环都执行一遍（单次往返）。假设所有超状态皆已扩展，该策略将得到一个针对缓存示例的测试案例。

替代性标准包括所有长度为 N 的路径，即所有给定长度的可能路径都必须被执行。但因为要编写更多测试案例造成的额外成本可能很高，所以不通过自动化工具一般很难达到这种标准。

至于状态对象，测试路径不会产生新的状态对象，因为状态已经存在了。但因为需要判定路径上的属性，状态对象中可能需要增加额外的检测方法。

返回

浏览器的返回按钮提供了需要特别注意的状态导航。虽然这个按钮在传统的超链接导航中是有意义的，但在如今的 Web 应用程序中，对于它到底应该具备什么行为并无定论。

网页开发者可以通过操纵历史堆栈来替换这个按钮的行为。正因如此，我们希望能够测试 Web 应用程序的返回按钮行为，而 WebDriver 为此提供了一个 API 调用。

对状态机来说，返回按钮不是一个独立的转换。相反，它是对先前（点击）事件的反应。正因如此，返回按钮的行为是边的一个属性，表示转换可以按照反方向“取消”。

根据 UML 的机制，为元素赋予特殊意义是用注释（轮廓）实现的。在图 13 中，明确的 <<back>> 和 <<noback>> 注释已经添加到了边上，用来表示返回按钮可否用于在点击后回到初始状态。因此，对于在“关于”、“正在注册”和“正在认证身份”状态之间的简单导航，返回按钮可以实现返回。

但在“已通过身份认证”和“正在认证身份”状态之间，返回按钮被有效地禁用了：一旦注销，任何人在点击“返回”后都不应该看到只有通过身份认证才能看到的内容。知道哪些转换有特殊返回行为后，我们就能更好地设计额外的测试场景来验证所需的行为。

带有历史记录的超状态

稍微复杂一些的超状态示例，设想一个可以根据每一列进行排序的表格。点击列标题可对表格排序，因而每一列都有一个子状态（图 14）。

这些事件来自于超状态，表示它们可以在任何子状态下被触发，随后进入另一个子状态。当离开可排序表格所在的页面时，例如请求关于指定行的详细信息时，我们需要决定，是设计成当返回该页面时（在此案例中是指点击“文件夹”链接）要产生一个按照默认列排序的表格（在此案例中指 A 列），还是设计成根据最后点击的列来恢复排序。

图 12.较长路径上的错误。

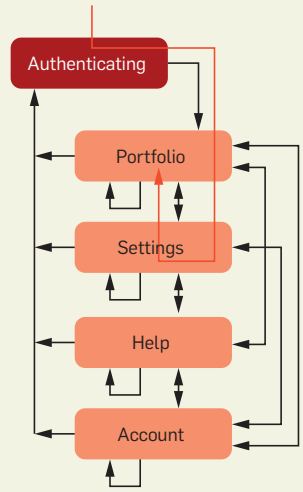


图 13.返回按钮注释。

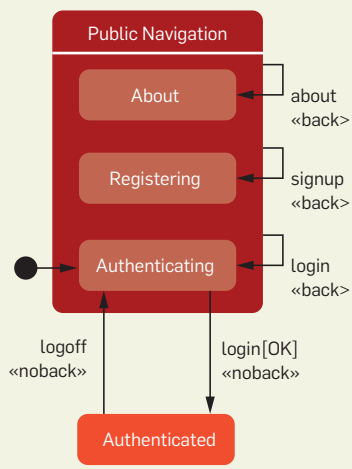


图 14.带有历史记录的超状态。

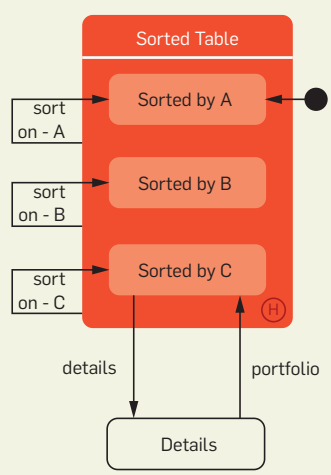


图 15.正交区域。

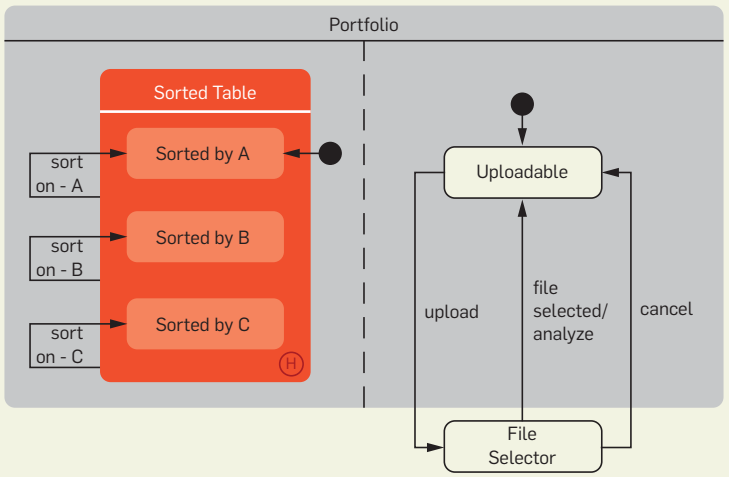


图 16.PhoneCat AngularJS 应用程序。

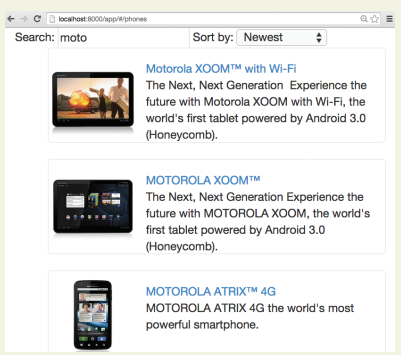
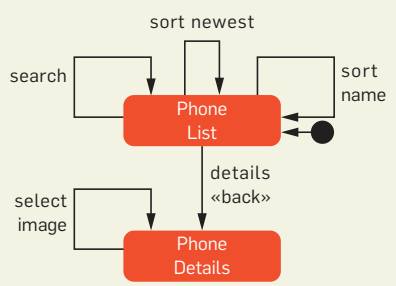


图 17.电话号码簿状态图。



在 UML 状态图中，第一种方案（返回超状态的初始状态）是默认的。第二种方案（返回超状态在转移前的状态）可以通过为超状态打上带圈的 H 标签、将其标记为历史状态来表示。在这两种情况下，如果这种行为很重要且需要测试，我们都需要一条额外路径（场景）来验证从非初始状态退出后，超状态能否返回正确的状态。

And 状态。如今的 Web 应用程序通常带有大量独立的小插件，例如在一个小插件中展示联系人列表，在另一个小插件中展示一组图片。这些小插件分别对应多个放置在同一页面的小型独立状态机。

在 UML 状态图中，这些状态可以用*正交区域*（又称 AND 状态）描述，如图 15 所示。该图展示的是“文件夹”的一种状态，包含一个可排序的表格和一个用来添加项目的上传按钮。它们可以单独使用，在图中体现为该“文件夹”状态被虚线切割成两半。该上传对话是模式对话，因此它位于“文件夹”类的外部。上传后，表格的排序保持不变，因此它被打上了 H 标签。

这样的正交区域可以用来表示出现在同一页面上的多个小型状态机。这些正交区域中的状态转换可来自用户交互。它们也可以通过服务器事件（通过 Web Socket）来触发，例如推送关于新邮件和股价调整的通知。

从测试的角度来说，正交区域原则上是独立的，因此也可以独立测试。

像 OR 状态一样，AND 状态也可以扩展，在这个案例中的目的是明确所有可能的组合情况。这将导致状态图变得极其臃肿，而且会大幅增加可能的测试案例数量。对这些组合情况中的一部分进行完整测试是有意义且可行的，但如果要全部测试，就必须依靠自动化测试手段。

基于状态的故事。最后同样重要的是，状态和状态图对通过用户故事和验收场景来描述需求也有帮助。例如，由技术顾问丹·诺斯 (Dan North)[9] 提出的故事格式就是一个很贴切的例子。这类故事包括一句以下这种形式的一般性描述“当……时，我希望……，以便……”，接下来是几种以下这种形式的验收场景“假设……，当……时，然后……”。

很多时候，这些验收场景可以直接映射为测试状态图中的单一转换。因而场景就可以用如下形式描述：

假设我已经到达某种状态
当我触发特定事件时
然后应用程序执行某个动作
并且应用程序转移到另一个状态。

这样，状态对象就允许 API 按照这些验收场景的描述与状态机交互。单个的测试案例是从一个状态转移到另一个状态，而完整的功能是通过一系列在状态机中状态转换的验收测试案例来描述的，同时会检测应用程序的行为。

AngularJS PhoneCat 示例

关于使用 WebDriver 和状态对象进行测试的完整示例，请参考 AngularJS PhoneCat 教程（https://docs.angularjs.org/tutorial）。图 16 显示了 PhoneCat 使用截图。它带有一个用 Protractor 编写的测试套件，包含在专为 AngularJS 应用程序打造的 WebDriverJS[11] 扩展中。

该应用程序含有一个简单的电话号码列表，可以按姓名筛选，也可以按字母或年龄排序。点击一个电话号码将显示给定电话类型的完整详情。

对于它的两种视图（电话号码列表和电话号码详情），该教程附带的 WebdriverJS 测试套件分别包

含三个测试案例，另外还包含一个针对入口 URL 的测试，因此共有七个测试案例。

原始教程中的测试套件没有使用页面（或状态）对象。为了演示状态对象的用法，我已经将 PhoneCat 测试套件改写成了一个基于状态对象的测试套件，该套件可以从我发布在 GitHub 上的 PhoneCat 软件版本分支获取（https://github.com/avandeursen/angular-phonecat/pull/1）。

PhoneCat 应用程序使用的状态图如图 17 所示。它指向两个状态对象（每种视图一个）。这些状态对象可以用来表达原始的各种场景集合。此外，该状态图还产生了额外的案例，例如原始测试案例中没有覆盖的“排序 - 最新”转换。

该图也清楚地表明，不能从电话号码详情直接进入电话号码列表。浏览器的返回按钮是交互设计明确的一部分，因此对应的转换上添加了 `<<back>>` 注释。（注意，这是唯一带有该属性的边：处于“电话号码列表”状态时，在其他任何转换后点击“返回”都将退出应用程序，这是 AngularJS 的缺省行为）。

鉴于返回按钮对于在两种视图之间导航至关重要，该基于状态的测试套件也通过一个场景描述了这种行为。

最后，因为 Protractor 和 WebDriverJS API 是完全基于异步 JavaScript Promise 的，[11] 所以状态对象的实现也是异步的。例如，“电话号码列表”状态对象提供了一种方法，这种方法会“执行一个命令来对电话号码列表排序”，而不会中断到电话号码排序完毕后。通过这样的方式，实际场景可以将所有 Promise 串起来，例如用 `then` Promise 操作符来串联。

AngularJS 的实际使用。本文展示的大部分插图是根据为一款 Web 应用程序制作的图设计的，该 Web 应用程序是为代尔夫特理工大学 (Delft University of Technology) 的一家衍生公司开发的。该应用程序允许用户注册、登录、上传文件、分析和将文件可视化，以及查看分析结果。

该应用程序的端到端测试套件使用了状态对象。它包含 25 个状态对象和 75 个场景。像 PhoneCat 测试套件一样，它也使用 Protractor，包含约 1,750 行 JavaScript 代码。

该端到端测试是在一台 TeamCity (https://www.jetbrains.com/teamcity/) 持续集成服务器上运行的，根据后端和前端的任何变化，共调用了约 350 个后端单元测试和所有的端到端场景。

该测试套件辅助修复和查找多个与客户端缓存、返回按钮行为、表格排序和图像加载相关的错误。这些问题之中有一些是由不正确的数据绑定造成的，例如 JavaScript 变量名称拼写错误，或更名重构不完整。这些测试还查出了一些后端 API 错误，有些错误与不正确的服务器配置和权限有关（该错误会导致 405 状态代码出现，偶尔还会导致 500 HTTP 状态代码出现），有些错误与不正确的客户端 / 服务器假设有关（服务器返回的 JavaScript Object Notation 不符合前端的预期）。

结论

对 Web 应用程序进行端到端测试时，可以利用状态来驱动测试：

- 将感兴趣的交互作为小型状态机来建模。
- 让每个状态对应一个状态对象。
- 在每个状态中包含一个自校验，来验证浏览器是否确实处于该状态。
- 对于每个转换，编写一个对原始和目标状态执行自校验的场景，并验证动作对转换的影响。
- 利用转换树来说明状态的可到达性和转换覆盖。
- 利用先进的状态图理念，例如 AND 状态、OR 状态和注释来让状态图保持简洁易懂。
- 考虑状态机中容易出错的特定路径；如果路径上的状态已有状态对象，那么测试路径上的行为应该不难。
- 在持续集成服务器上运行端到端测试套件，来定位 HTML、JavaScript 和后端服务之间的集成问题。

就页面对象而言，浏览器交互的详细信息会被封装在状态对象内部，因而对测试场景是透明的。最重要的是，状态图和对应的状态对象可以为测试套件设计的整体流程中提供直接指导。

queue.acm.org 上的相关文章

Rules for Mobile Performance Optimization
Tammy Everts
http://queue.acm.org/detail.cfm?id=2510122

Scripting Web Service Prototypes
Christopher Vincent
http://queue.acm.org/detail.cfm?id=640158

Software Needs Seatbelts and Airbags
Emery D. Berger
http://queue.acm.org/detail.cfm?id=2333133

参考资料

1. AngularJS. ngInclude directive; https://docs.angularjs.org/api/ng/directive/ngInclude.
2. Antoniol, G., Briand, L.C., Di Penta, M. and Labiche, Y. A case study using the round-trip strategy for state-based class testing.In *Proceedings 13th International Symposium on Software Reliability Engineering*.IEEE (2002), 269–279.
3. Binder, R.V. *Testing Object-oriented Systems*.Addison-Wesley, Reading, PA, 1999, Chapter 7.
4. Fowler, M. PageObject, 2013; http://martinfowler.com/bliki/PageObject.html.
5. Harel, D.1987.Statecharts: a visual formalism for complex systems.*Science of Computer Programming* 8(3):231–274.
6. Horrocks, I. *Constructing the User Interface with Statecharts*.Addison-Wesley, Reading, PA, 1999.
7. Leotta, M., Clerissi, D., Ricca, F., Spadaro, C. Improving test suites maintainability with the page object pattern:An industrial case study.In *Proceedings of the Testing:Academic and Industrial Conference— Practices and Research Techniques*. IEEE (2013), 108–113.
8. Mesbah, A., van Deursen, A. and Roest, D. Invariant-based automatic testing of modern Web applications. *IEEE Transactions on Software Engineering* 38, 1 (2012) 35–53.
9. North, D. What’s in a story?; http://dannorth.net/whats-in-a-story/.
10. Selenium.Page Objects, 2013; https://github.com/SeleniumHQ/selenium/wiki/PageObjects
11. Selenium.Promises.In WebDriverJS User’s Guide, 2014; https://code.google.com/p/selenium/wiki/WebDriverJs#Promises.
12. SeleniumHQ.WebDriver; http://docs.seleniumhq.org/projects/webdriver/.

鸣谢

感谢 Michael de Jong、Alex Nederlof 及 Ali Mesbah 参与的大量有益讨论和他们为本文提供的反馈。本文使用的 UML 图是用免费的 UML 绘图工具 UMLet（版本号 13.2）绘制的。

Arie Van Deursen 是代尔夫特理工大学 (Delft University of Technology) 的教授，担任软件工程研究小组带头人。为了将他的研究成果付诸实践，他于 2000 年与他人共同创立了 Software Improvement Group，并于 2010 年创立了 Infotron 公司。

译文责任编辑：陈文光；校对者：白晓颖

DOI:10.1145/2699415

Quipper 语言为量子计算提供了一个统一的通用编程框架。

作者：BENOÎT VALIRON、NEIL J. ROSS、PETER SELINGER、D. SCOTT ALEXANDER 以及 JONATHAN M. SMITH

量子编程的未来

世界上最早的电脑（如 ENIAC）不仅非常罕见，而且极难编程。因为此类电脑要求用以表达算法的“词汇”必须适合现有特定硬件，如：ENIAC 电脑的函数表，以及后来电脑上更为常规的算术和移动操作。以 FORTRAN 语言为代表的符号编程语言问世后，算法可以用更适合人类理解的方式得以规范，并将规范转化为可由电脑执行的形式，为新一代计算设备解决了一大难题。用于此种规范的“编程语言”缩小了人和计算装置之间的语义差距。它提供了两个重要的功能：高度抽象（用以自动记账）和模块化（使得程序各子部分的推出更为容易）。

量子计算是一种计算模式，它将数据编码在遵守量子物理学定律的对象状态上。运用量子技术设计的算法可能优于相应的最好的常规或传统算法。

量子计算机的设想早在 20 世纪就已提出，其很可能将在 21 世纪成为现实，并从实验室走向商用领域。这样，从传统计算设备编程中所获得的诸多经验教训，就可以运用于新兴的量子计算设备。

量子协处理器模型

编程人员将如何与可进行量子运算的设备进行交互？本文的目的并非为制造现实的量子计算机提供工程蓝图。Meter 和 Horsman [13] 已探讨过这一计划，请查询相关著述。本文详细描述了一个假想的量子体系结构，足以探讨编程人员将如何为其编程。

从外部看，量子计算机执行一套专门的操作，有些类似于浮点单元或图形协处理器。因此，本文把量子计算机设想为由传统计算机控制的一种协处理器，如图 1 所示。传统计算机为量子协处理器执行各种操作，如：编译，常规记账，正确性检查，以及代码和数据的制作。

» 重要见解

- **量子计算机科学是一门全新学科，研究的是量子计算在各方面的实用集成问题，从研究论文中的抽象算法到物理运算都涵盖在内。**
- **以量子编程语言编写的程序应尽量接近于非正式的高级描述，不包含适用于量子协处理器模型的输出。**
- **量子编程环境的其他重要方面包括，在部署前自动估算离线资源，为测试、规范和验证提供便利。**

插图由 PETER BOLLINGER 提供

量子协处理器只执行量子变换，如初始化，幺正运算和测量。这种量子计算机模型即 Knill 所提出的 QRAM 模型 [11]，学界认为该模型最有可能最后发展为现实中的量子计算机 [13]。

某些硬件密集型低层控制运算（如量子误差校正）可直接整合到量子协处理器。本文假设，量子协处理器配有一个高速专门固件，进行低级“量子运行时”。量子固件针对每个实际制造的量子协处理器量身定制，并在协处理器外单独编程。虽然量子固件高度依赖于特定硬件的物理规格，但其独立于要运行的算法。

图 1. 量子协处理器模型中的混合计算。

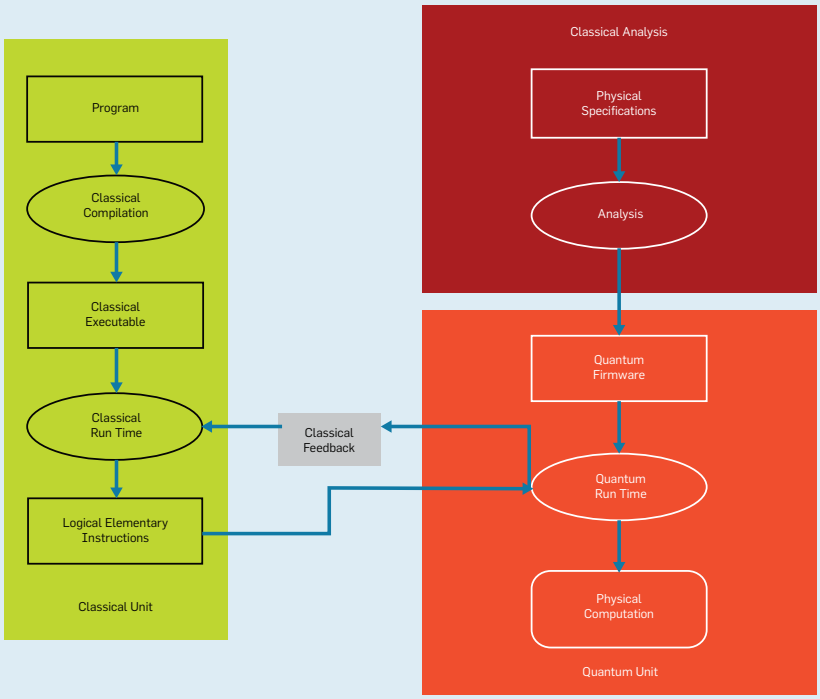


图 2. 量子电路

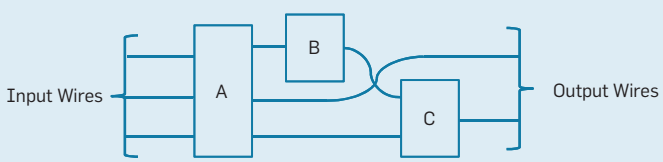
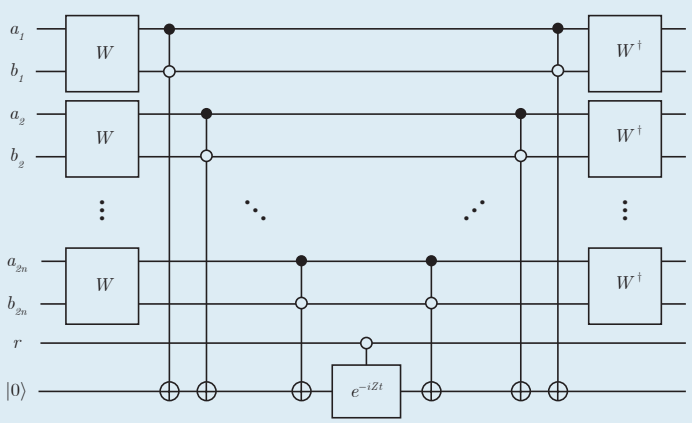


图 3. 量子电路片段



任何量子程序的源代码都位于传统单元上。通过常规的传统编译，源代码生成将在传统计算机上运行的可执行代码。传统计算机能够在一个信息队列上发送基本指令（如：“分配一个新的量子位”、“旋转量子位 x”和“测量量子位 y”），量子协处理器通过该信息队列与传统控制器进行交互。量子协处理器完成一次操作后，传统计算机能通过该信息队列读取运算结果。该模型的算法控制流是传统的算法控制流，测试和循环都在传统计算机上执行。传统数据和量子数据都是第一类对象。

传统计算机运行时通过信息队列获取量子协处理器的反馈，如测量的结果。根据该算法，这种反馈可能仅发生在量子计算（批处理模式操作）快结束时，或在传统计算机正发送基本指令时（在线运算）。因为可能涉及在线运算，这就带了更多工程挑战，控制量子协处理器的传统控制器运算速度必须要足够快，能够与量子运行时进行实时交互。另外，许多常见的量子算法仅要求批处理模式运算。本文假设存在一个足够灵活的量子编程模型，能解决两种类型的运算。

在传统编程环境下，数据的逻辑结构与其在硬件上的物理表现形式被分开。在本文提出的模型中，虽然算法是在逻辑层实施的，但量子位被编码在物理硬件上。这一任务由编译程序和量子固件完成，它们把逻辑量子位和运算映射到稳定的物理表现形式，并进行适当的误差校正。

描述量子算法

为了激发人们对表达力较强的量子编程语言 (QPL) 的需要，本文简要地考虑了文献已有的量子算法通常的规范方式。量子算法一般由传统运算和量子操作组合而成。算法的量子部分通常聚合到量子电路中。其中，量子门表示为方框，量子位表示为导线，如图 2 所示。量子电路存在局限性。例如，它们不能包含循环，所以导线必须朝向同一个方向。量子门可以有多个输入端和输出端。除了与量子位生成和破坏（测量）相对应的的量子门，其他基本运算始终是么正变换，因而它们必须具有相同数量的输入端和输出端。

量子算法的典型描述包括以下因素中的一个或多个，依不同层次的形式而定。

*数学方程式。*数学方程式可用于描述态制备、么正变换和测量。例如，Harrow 等人 [9] 描述了一个可用来求解线性方程组的量子算法。该算法的一个子电路定义如下：

$$\sum_{t=1}^{T}\begin{pmatrix}|t{+}1\rangle\langle t|\otimes U_t \\ +|t{+}T{+}1\rangle\langle t{+}T|\otimes I \\ +|t{+}2T{+}1 \bmod 3T\rangle\langle t{+}2T|\otimes U_{3T+1-t}^{\dagger}\end{pmatrix}$$

*已知量子子例程的调用。*例如，量子傅里叶变换，相位估计，振幅放大，以及随机游动。Harrrow 等人的算法要求 [9] “使用相位估计在特征向量基底上分解 $|b\rangle$”；

Oracle。（神谕）这些是传统可计算函数，必须可逆并编码为量子运算。它们常在非常高的层次被描述。如 Burham 等人 [3] 定义了这样的一个 Oracle: “语句的真值” $f(x) \leq f(y)$。

图 4.反向和重复

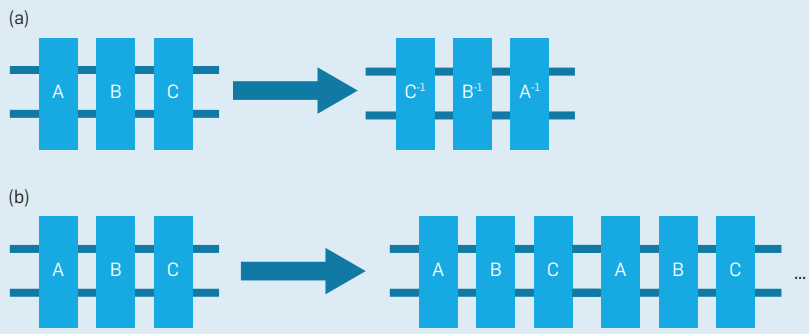


图 5.初始化-运行-测量循环

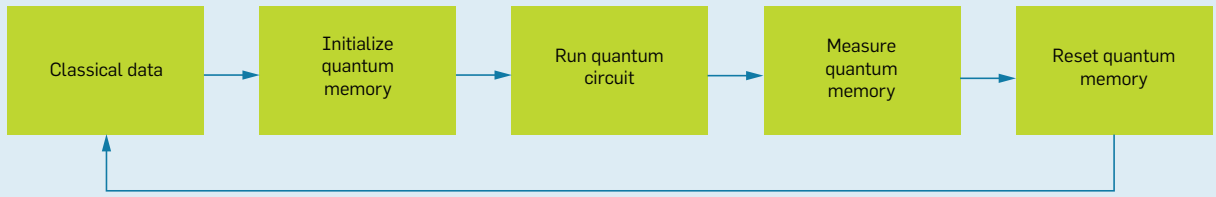


图 6.从中间测量获得反馈的电路

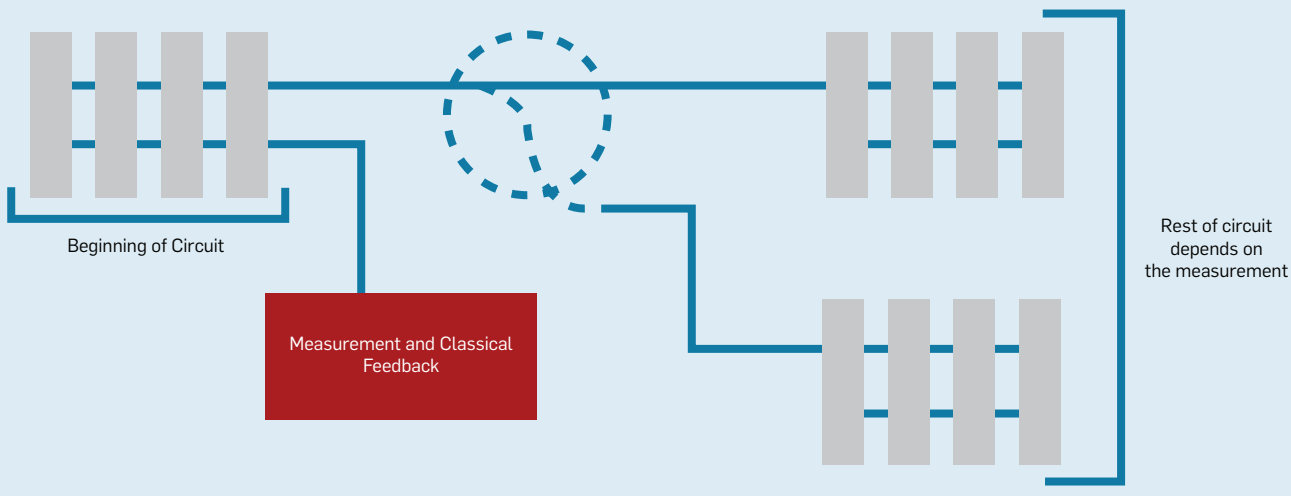
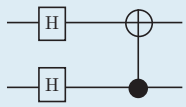


图 7.过程化示例。

```
mycirc ::Qubit -> Qubit -> Circ (Qubit, Qubit)
mycirc a b = do
  a <- hadamard a
  b <- hadamard b
  (a,b) <- controlled_not a b
  return (a,b)
```



电路片段。例如，图 3 所示的电路由 Childs 等人[4]设计。严格来说，该图描述了*一组*量子电路，其参数为旋转角度 t 和尺寸参数 n，在非正式电路中用省略号“…”标示。在正式电路中，这种参数相关性必须明确说明。

电路的高层次运算。例如，“反向”（即逆转电路）和“迭代”（即重复电路），如图 4 所示；以及

传统控制：许多算法涉及传统单元和量子单元之间的交互。这种相交互可以采取迭代的形式，如图 5 所示，反复重新运行相同的量子电路；也可以采取反馈的方式，如图 6 所示，根据先前测量的结果，快速生成量子电路。

量子编程语言的要求

理想情况下，量子编程语言应允许编程人员在抽象层上实现量子算法，这基本符合人们对算法的认识。如果算法以数学公式描述最自然，则编程语言应尽可能支持此类描述方法。同样，如果算法以低级量子门序列描述最自然，则编程语言应尽可能支持这种描述方法。

因此，文献中用于表示诸多算法的标准方法，可作为编程语言的设计准则。Knill [11] 提出了量子编程的要求，包括：

分配和测量。使得分配和测量量子寄存器及进行幺正运算成为可能；

推理子例程。允许推理和倒置程序量子子例程，并在调节子程序在一个量子寄存器的状态；以及

构建量子 oracle。使程序员能够根据对传统函数的描述构建量子 oracle。

根据我们在实现[a]量子算法方面的经验，一门量子编程语言需要具有以下其他特征：

量子数据类型。在传统编程语言中，数据类型的作用是允许编程人员抽象地思考数据，而不是管理比特和字。例如，在大多数情况下，浮点数最好视为支持特定算法运算的基本数据类型，而不应视为组成指数、尾数和符号的 64 位数组。同样，许多量子算法指定了更为丰富的数据类型，所以量子编程语言也应提供相应的抽象。例如，量子线性系统算法 (Quantum Linear System)[9] 需要操作量子整数以及量子实数和复数，这些数据类型可通过浮点或固定精度编码表示。又如 Hallgren[8] 为计算实二次数域类数而设计的算法。此算法涉及的一种必须置于量子叠加态的数据类型就是代数数域的理想；

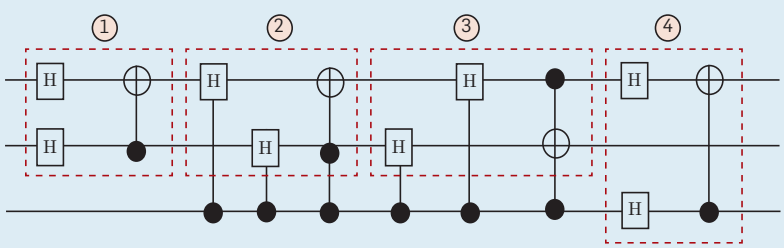
规范和验证。在传统编程领域，验证程序正确性的技术有很多，包括编译时类型检查，运行时类型检查，形式化验证，以及调试。其中，形式验证可以说是最可靠但最昂贵的技术。强编译时保证的可用性，需要编程语言的设计必须非常仔细。调试是既便宜又实用的技术，因而在传统程序开发中应用广泛。

在量子计算领域，调试的成本可能很高。首先，观测量子系统会改变它的态。在现实的量子硬件上运行时，量子程序的调试器提供的有关态的信息必然不完整。另一种方法是使用量子模拟器进行调试。但是，由于模拟量子系统的指数量级的成本开销，这种方法不太实用。此外，可以预期，早期的量子计算机将会非常罕见且运行成本也会较高，因而早期量子码出现运行时错误的成本将比传统计算机要高

a 我们所说的“实现”一个算法，是指将该算法作为计算机程序实现，这并不代表我们已经真正在量子计算机上运行了程序，但我们在量子模拟器上运行了算法的部分内容。

图 8.块结构示例。

```
mycirc2 ::Qubit -> Qubit -> Qubit -> Circ (Qubit, Qubit, Qubit)
mycirc2 a b c = do
(1) mycirc a b
    with_controls c $ do
      mycirc a b
(2)   mycirc b a
(3) mycirc a c
(4) return (a,b,c)
```



得多。因而，人们不采用性本较低的量子编程分析，而使用严格的编译时正确性保证，以及形式规范和验证。

量子编程语言应具备一套健全且严格定义的语义，实现程序行为的数学规范和程序正确性证明。另外，量子编程语言还应具备完善的静态类型系统，能防范绝大多数运行时错误（如违反量子信息不可克隆性）；[b] 以及

资源敏感性和资源评估。首先，量子计算机将可能不会有很多的量子位。因此，量子编程语言应包含专门的工具，可以在部署前评估运行某特定代码（如量子位数目、初等量子门数目及其他相关资源的数目）所需的资源。量子编程语言设计者们还需解决一个特别问题：如何处理量子纠错。随着该领域不断取得进步，设计者们必须决定哪种纠错方式效率更高：让程序员进行纠错（可能通过手工优化）或者让编译器或一些其它工具进行自动纠错。因为量子纠错会增加冗余，而编译器的优化措施会消除冗余，两者可能进行不良互动，所以任何量

b 非克隆性可从物理层面得到保证，与编程语言无关。同理，禁止非法内存访问可以通过传统处理器的缺页机制得到保证。但是，最好也有一种编程语言能够保证，*在运行程序以前*编译的程序永远都不会尝试访问非法内存位置，不会尝试将受控非门应用到量子位 n 和 m（其中 $n = m$）。

图 9. 电路运算符示例。

```
timestep ::Qubit -> Qubit -> Qubit -> Circ (Qubit, Qubit, Qubit)
timestep a b c = do
    mycirc a b
    qnot c `controlled` (a,b)
    reverse_simple mycirc (a,b)
    return (a,b,c)
```

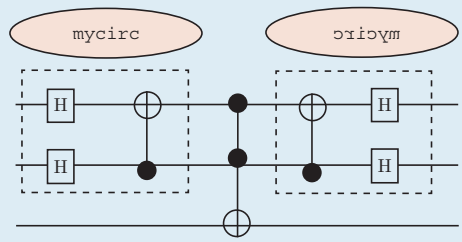
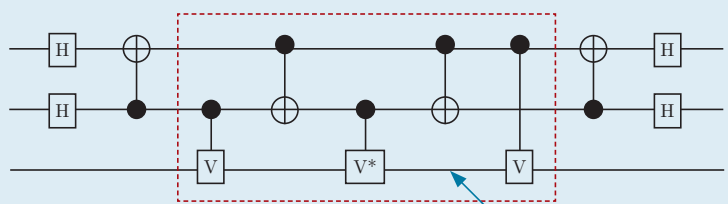


图 10.电路转换器示例。

```
timestep2 ::Qubit -> Qubit -> Qubit -> Circ (Qubit, Qubit, Qubit)
timestep2 = decompose_generic Binary timestep
```

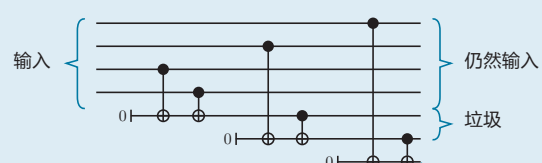


Toffoli 门的二元量子门分解

图 11.从函数到逆向的转化的示例。

```
build_circuit
f :: [Bool] -> Bool
f as = case as of
  [] -> False
  [h] -> h
  h:t -> h `bool_xor` f t

unpack template_f :: [Qubit] -> Circ Qubit
```



输出

图 12. 图 11 电路的逆向形式。

```
classical_to_reversible ::(Datable a, QCData b) => (a -> Circ b) -> (a,b) -> Circ (a,b)
classical_to_reversible (unpack template_f)
```

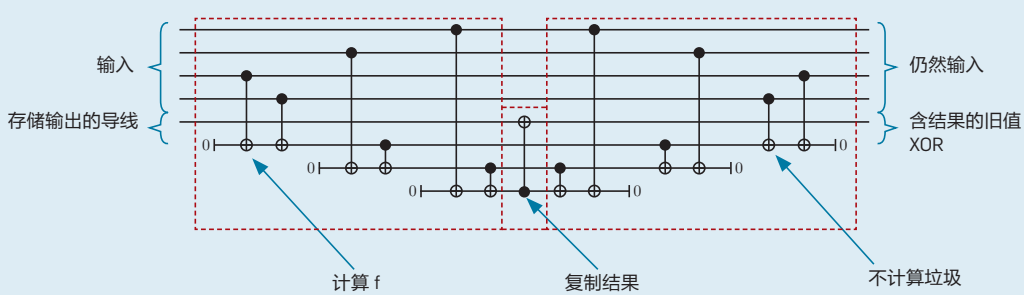


图 13.过程化示例。

```
   Quipper

w ::(Qubit,Qubit) -> Circ (Qubit,Qubit)
w = named_gate "W"

toffoli ::Qubit -> (Qubit,Qubit) -> Circ Qubit
toffoli d (x,y) =
   qnot d `controlled` x .==.1 .&&. y .==.0

eiz_at ::Qubit -> Qubit -> Circ ()
eiz_at d r =
   named_gate_at "eiZ" d `controlled` r .==.0

circ :: [(Qubit,Qubit)] -> Qubit -> Circ ()
circ ws r = do
   label (unzip ws,r) (("a","b"),"r")
   with_ancilla $ \d -> do
   mapM_ w ws
   mapM_ (toffoli d) ws
   eiz_at d r
   mapM_ (reverse_generic w) (reverse ws)
   return ()

main = print_generic EPS circ (replicate 3 (qubit,qubit)) qubit
```

子编程语言的设计和运用必须注意量子纠错的要求。

QOL 的先前研究

世界各地的研究人员发明了多种量子编程语言。[5]van Tonder 的量子 λ 演算等量子编程语言 [18] 主要用作理论工具。Ömer 的 QCL 可以说是首个专为实际应用设计的量子编程语言，[14] 它是一种支持“结构量子编程”的 C 语言类型的命令式语言。QCL 提供了简单的寄存器，但不具备高级的量子数据类型。它对规范和验证的支持更强大。Bettelli 等人 [2] 借鉴了 Ömer 的部分研究，设计了一种量子编程语言，它是对 C++ 语言的扩展。Sanders 和 Zuliani[16] 设计的 qGCL 卫式命令语言为程序规范语言的发明提供了灵感。

图 14.由图 13 中的代码生成的电路，带有 3 对量子位

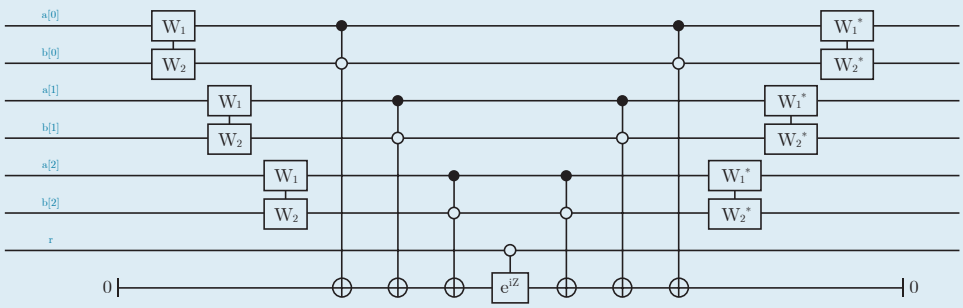
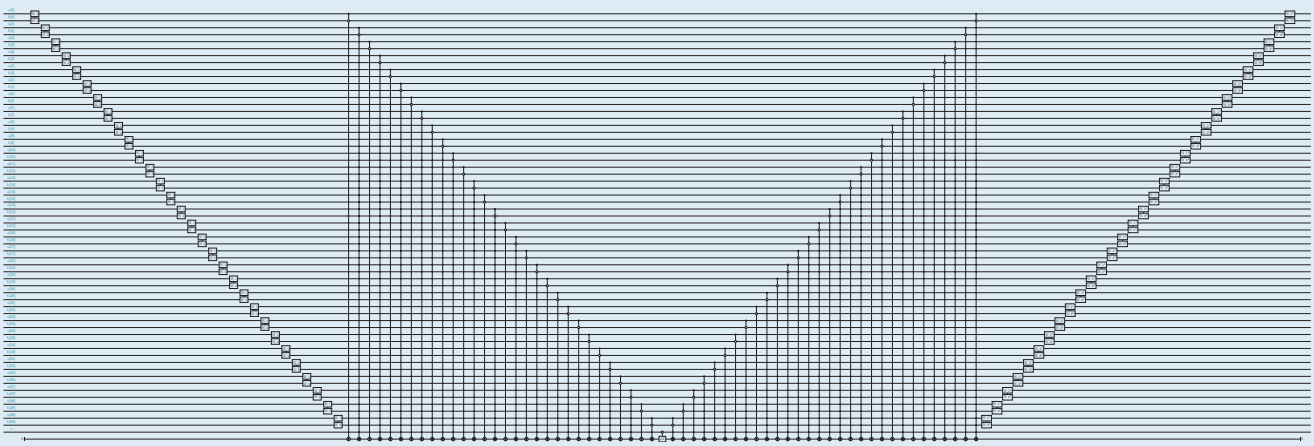


图 15.由图 13 中的代码生成的电路，带有 30 对量子位



Selinger 和 Valiron 设计的量子 λ 演算是第一门函数式量子编程语言[17]，它为操作传统数据和量子数据提供了一个统一的框架。该量子 λ 演算有明确的数学语义，可以确保类型正确的程序不会出现运行时错误。该语言可以利用归纳数据类型（如列表）和递归轻松地扩展。但一个缺点就是无法将电路构建和电路求值分离开来。因此，该语言无法将量子电路作为数据操作，也无法从传统描述自动构建幺正电路。由 Green 和 Altenkirch 设计的 Quantum IO Monad 解决了部分问题，该函数语言正是 Quipper 的鼻祖。[7]

图 16. calcRweights 函数。

```
calcRweights y nx ny lx ly k theta phi =
     let (xc',yc') = edgetoxy y nx ny in
     let xc = (xc'-1.0)*lx - ((fromIntegral nx)-1.0)*lx/2.0 in
     let yc = (yc'-1.0)*ly - ((fromIntegral ny)-1.0)*ly/2.0 in
     let (xg,yg) = itoxy y nx ny in
     if (xg == nx) then
        let i = (mkPolar ly (k*xc*(cos phi)))*(mkPolar 1.0 (k*yc*(sin phi)))*
                ((sinc (k*ly*(sin phi)/2.0)) :+ 0.0) in
         let r = ( cos(phi) :+ k*lx )*((cos (theta - phi))/lx :+ 0.0) in i * r
     else if (xg==2*nx-1) then
         let i = (mkPolar ly (k*xc*cos(phi)))*(mkPolar 1.0 (k*yc*sin(phi)))*
                ((sinc (k*ly*sin(phi)/2.0)) :+ 0.0) in
       let r = ( cos(phi) :+ (- k*lx))*((cos (theta - phi))/lx :+ 0.0) in i * r
     else if ( (yg==1) && (xg<nx) ) then
         let i = (mkPolar lx (k*yc*sin(phi)))*(mkPolar 1.0 (k*xc*cos(phi)))*
                ((sinc (k*lx*(cos phi)/2.0)) :+ 0.0) in
       let r = ( (- sin phi) :+ k*ly )*((cos(theta - phi))/ly :+ 0.0) in i * r
     else if ( (yg==ny) && (xg<nx) ) then
         let i = (mkPolar lx (k*yc*sin(phi)))*(mkPolar 1.0 (k*xc*cos(phi)))*
                ((sinc (k*lx*(cos phi)/2.0)) :+ 0.0) in
          let r = ( (- sin phi) :+ (- k*ly) )*((cos(theta - phi)/ly) :+ 0.0)
in i * r
     else 0.0 :+ 0.0
```

硬件独立性。Quipper 的模式是在逻辑电路层查看量子计算。添加纠错代码和向硬件映射这两项任务由编译链下游的其他组件完成。

扩展的电路模式。为了附属管理的目的，量子位的初始化和终止都被明确跟踪；

层次型电路。Quipper 在电路或“框式子电路”层级设有子例程，可以在内存中紧凑地表示电路。

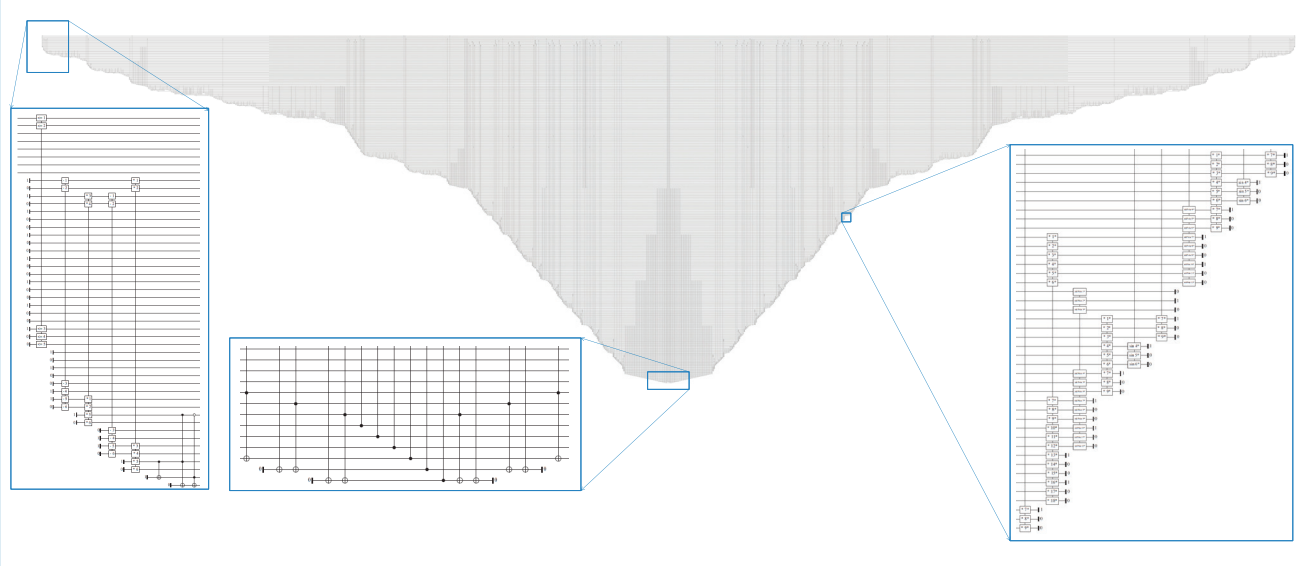
多样的电路描述语言。Quipper 支持多种编程类型，可以处理计算的过程化模式和函数模式。它还支持使用可编程操作对电路进行高级操作。

两个运行时。Quipper 程序通常描述依赖于一些传统参数的一组电路。第一个运行时为“电路生成”，

Quipper 语言

在前人研究的基础上，我们推出了用于量子计算的函数式语言 Quipper。我们选择将 Quipper 实现为主语言 Hskell 内的深层嵌入域专用语言 (EDSL)；请参见 Gill[6] 的著述以了解 EDSL 及其在 Haskell 的嵌入情况。Quipper 旨在为量子计算提供了一个统一的通用编程框架。其主要功能包括：

图 17. calcRweights 电路。



第二个运行时为“电路执行”。如前文所述，在批处理模式操作中，这两个运行时会相继发生，而在在线操作中，二者可能重叠发生；

参数/输入区分。Quipper有两种传统数据概念：“参数”，必须在电路生成时获知；“输入”，只能在电路执行时获知。例如，`Bool`类型用于布尔参数，`Bit`类型用于布尔输入；

可扩展数据类型。Quipper使用Haskell强大的类型-类别机制，提供抽象且可扩展的量子数据视图。

自动生成量子oracle。很多量子算法都要求一些重要的传统计算是可逆的，然后才被提升至量子运算。Quipper具备将普通的Haskell程序变成可逆电路的工具。该功能是借助Haskell的“Template Haskell”实现的，这是一项自引用功能，可以检查和操作自己的源代码。

Quipper功能亮点

我们将重点介绍Quipper的一些功能，并搭配代码示例：

过程化模式。在Quipper语言中，量子位包含在变量中，量子门每次只应用一个。电路生成函数的类型是由箭头后的关键字`Circ`确定的，如图7所示。函数`mycirc`输入了类型为`Qubit`的输入`a`和`b`，并在生成电路的同时输出了一对量子位；

块结构。生成电路的函数可以作为子例程重复使用，进而生成更大的电路。运算符（例如`with_control`）可以将整个代码块用作参数，如图8所示。请注意，“`do`”引入了缩进式代码块，“`$`”是Haskell语法中特有的，本文读者可以忽略。

电路运算符。Quipper将电路作为数据处理，提供可操作整个电路的高级运算符。例如，运算符`reverse_simple`可以逆转电路，如图9所示。

电路转换器。Quipper提供了用户可编程的“电路转换器”，作为按量子门修改电路的机制。以图9中的`timestep`电路为例，使用`Binary`转换器可以将其分解为二元量子门。

从函数到逆向自动转化。Quipper提供了特殊关键字`build_circuit`，可以自动从普通函数程序合成电路，参见图11。生成的电路可以使用运算符`classical_to_reversible`实现逆转，参见图12。

Quipper的使用经验

我们根据美国情报先进研究计划署(Intelligence Advanced Research Projects Activity (IARPA)量子计算机科学项目提供的文件，已经使用Quipper实现了七个重要的量子算法。[10]所有这些算法在某种意义上都可以运行，即我们可以打印相应的小参数电路，对难以处理的小规模电路自动计算量子门数量。这些算法（参见表格）分别解决了不同的经典难题，每种算法都提供了渐进的量子加速，尽管不一定是指数级的加速。这七种算法涵盖了量子技术的众多方面，例如，表格中的数个算法都使用量子傅里叶变换、相位估计、Trotterization以及振幅放大。IARPA专门选择相对复杂的算法，因此一些著名但技术上简单的算法（例如Shor的因子分解算法）没有入选。

我们使用Quipper可以为每种算法执行半自动或全自动逻辑门数量估算，甚至是很大规模的问题。例如，在查找三角形的算法中，

```
命令 ./tf -f gatecount
-o orthodox
-n15 -l31 -r6
```

会生成整个算法的门数量，并显示在2^{15}个顶点的图（使用包含31位整数的oracle）和元组大小为2^6的Hamming图上。该命令从运行到完成在笔记本电脑上用时不到2分钟，可以为该算法实例计算出30,189,977,982,990个量子门和4,676个量子位。

示例。为作进一步说明，以下是两个以Quipper语言编写的子例程：

过程化示例。首先，我们确定图3中的电路组。该电路实现了二进制结合树(Binary Welded Tree)算法中的量子游走时间步骤。[4]电路输入了一列量子位对（a_i, b_i）和一个量子位r。它先生成了态为$|0\rangle$的附属或暂存空间量子位。然后将两个量子位的量子门W应用到每对(a_i, b_i)，又将一系列双重受控非门应用到附属量子位。在中间门e^{iZt}之后，又以相反顺序应用所有的门。附属量子位以态$|0\rangle$结束，并且不再需要。Quipper代码在图13中，

所选的量子算法。

算法	说明
二进制结合树 (Binary Welded Tree)[4]	通过执行量子游走在图中找到标记的节点
布尔表达式 (Boolean Formula)[1]	使用量子模拟评估指数级的大型布尔表达式；QCS 版本计算六贯棋游戏的制胜策略。
类数 (Class Number)[8]	估算实二次数域类数
基态估算 (Unique Shortest Vector)[19]	计算分子的基态能量级别
量子线性系统 (Quantum Linear Systems)[9]	解决大型而稀疏的线性方程组
唯一最短向量 (Unique Shortest Vector)[15]	在n维的方格中找到最短向量
三角形查找 (Triangle Finding)[12]	在大型密集图中找到唯一的三角形

可以生成图 14 中的电路。如果在主函数中以“30”代替“3”，就可以得到此电路组的更大实例，如图 15 所示。

*从函数到逆向的转化。*此示例来自量子线性系统算法 (Quantum Linear System)。[9] 除其他组成部分外，该算法还包含一个用来计算复数向量 *r* 的 oracle，图 16 显示了该 oracle 的核心函数。请注意，它高度依赖于真实复数的代数运算和超越运算（例如 `sin`、`cos`、`sinc` 以及 `mkPolar`）以及图 16 中未显示的子例程（例如 `edgetoxy` 和 `itoxy`）。此函数可以使用 Quipper 的电路自动生成工具直接处理。代数函数和超越函数自动映射到由现有的 Quipper 定点真实复数算术库提供的量子版本。结果如图 17 中的大型电路。

结论

实用量子计算需要使用从抽象算法描述到物理粒子层面的工具链。量子编程语言是此工具链中的重要组成部分。在理想情况下，此类语言应支持量子算法以高度抽象的形式表达，就像研究论文中所描述的那样，并且此类语言还应能够将量子算法转换为逻辑电路。我们将此逻辑电路看做一种中间表达形式，它可由其他工具进一步处理，即添加量子控制和错误纠正，最终传递到控制物理运算的实时系统。

Quipper 是适用于量子协处理器模型的语言之一。我们通过实现数个大规模算法证明了 Quipper 的可行性。Quipper 的设计解决了与量子计算相关的一些重要编程语言难题，但仍有很多工作要做，特别是设计适用于量子计算的类型系统。Quipper 作为一种嵌入式语言，仅能在 Haskell 的类型系统中应用，从而提供很多重要的安全保障。但是，由于 Haskell 缺少对线性类型的支持，一些安全属性（例如不尝试克隆量子信息）无法得到充分支持。为量子计算设计更好的类型系统将成为今后的研究主题。 C

参考资料

1. Ambainis, A., Childs, A.M., Reichardt, B.W., Špalek, R., and Zhang, S. Any AND-OR formula of size *N* can be evaluated in time $N^{1/2+o(1)}$ on a quantum computer.*SIAM Journal on Computing 39*, 2 (2010), 2513–2530.
2. Bettelli, S., Calarco, T., and Serafini, L. Toward an architecture for quantum programming.*The European Physical Journal D 25*, 2 (2003), 181–200.
3. Burham, H., Durr, C., Heiligman, M., Hoyer, P., Magniez, F., Santha, M., and de Wolf, R. Quantum algorithms for element distinctness.In *Proceedings of the 16th Annual IEEE Conference on Computational Complexity* (Chicago, June 18–21).IEEE Computer Society Press, 2001, 131–137.
4. Childs, A. M., Cleve, R., Deotto, E., Farhi, E., Gutmann, S., and Spielman, D.A.Exponential algorithmic speedup by a quantum walk.In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing* (San Diego, CA, June 9–11).ACM Press, New York, 2003, 59–68.
5. Gay, S.J.Quantum programming languages:Survey and bibliography.*Mathematical Structures in Computer Science 16*, 4 (2006), 581–600.
6. Gill, A. Domain-specific languages and code synthesis using Haskell.*Commun.ACM 57*, 6 (June 2014), 42–49.
7. Green, A. and Altenkirch, T. The quantum IO monad. In *Semantic Techniques in Quantum Computation*, S. Gay and I. Mackie, Eds.Cambridge University Press, Cambridge, U.K., 2009, 173–205.
8. Hallgren, S. Polynomial-time quantum algorithms for Pell’s equation and the principal ideal problem. *Journal of the ACM 54*, 1 (Mar. 2007), 4:1–4:19.
9. Harrow, A.W., Hassidim, A., and Lloyd, S. Quantum algorithm for solving linear systems of equations. *Physical Review Letters 103*, 15 (Oct. 2009), 150502-1–150502-4.
10. IARPA Quantum Computer Science Program. *Broad Agency Announcement IARPA-BAA-10-02*, Apr. 2010; https://www.fbo.gov/notices/637e87ac1274d030ce2ab69339ccf93c
11. Knill, E.H.*Conventions for Quantum Pseudocode. Technical Report LAUR-96-2724.*Los Alamos National Laboratory, Los Alamos, NM, 1996.
12. Magniez, F., Santha, M., and Szegedy, M. Quantum algorithms for the triangle problem.*SIAM Journal on Computing 37*, 2 (2007), 413–424.
13. Meter, R.V. and Horsman, C. A blueprint for building a quantum computer.*Commun.ACM 56*, 10 (Oct. 2013), 84–93.
14. Ömer, B. *Quantum Programming in QCL*.Master’s Thesis.Institute of Information Systems, Technical University of Vienna, Vienna, Austria, 2000; tph.tuwien.ac.at/~oemer/qcl.html
15. Regev, O. Quantum computation and lattice problems. *SIAM Journal on Computing 33*, 3 (2004), 738–760.
16. Sanders, J.W. and Zuliani, P. Quantum programming. In *Proceedings of the Fifth International Conference on Mathematics of Program Construction, Vol. 1837 of Lecture Notes in Computer Science* (Ponte de Lima, Portugal, July 3–5).Springer-Verlag, Berlin Heidelberg, 2000, 80–99.
17. Selinger, P. and Valiron, B. A lambda calculus for quantum computation with classical control. *Mathematical Structures in Computer Science 16*, 3 (2006), 527–552.
18. van Tonder, A. A lambda calculus for quantum computation.*SIAM Journal of Computation 33*, 5 (2004), 1109–1135.
19. Whitfield, J.D., Biamonte, J., and Aspuru-Guzik, A. Simulation of electronic structure Hamiltonians using quantum computers.*Molecular Physics 109*, 5 (Mar. 2011), 735–750.

Benoît Valiron (benoit.valiron@monoidal.net) 是巴黎中央理工-高等电力学院工程学院的助理教授，也是法国巴黎第十一大学计算机科学实验室的研究员。

Neil J. Ross (neil.jr.ross@gmail.com) 是加拿大新斯科舍省哈利法克斯市的戴尔豪斯大学的博士生。

Peter Selinger (selinger@mathstat.dal.ca) 是加拿大新斯科舍省哈利法克斯市的戴尔豪斯大学的数学教授。

D. Scott Alexander (salexander@appcomsci.com) 是纽泽西州巴斯金里奇市应用传播学实验室的首席科学家。

Jonathan M. Smith (jms@cis.upenn.edu) 是宾夕法尼亚州费城的宾夕法尼亚大学的工程和应用科学教授以及计算机和信息科学教授。

译文责任编辑：孙晓明

观看此独家《通讯》视频中作者对本研究的讨论：http://cacm.acm.org/videos/exascale-computing-and-big-data

DOI:10.1145/2699416

本文探索了三大跨学科领域和它们重叠的程度。它们是否都属于同一个更大的范畴?

作者：THANASSIS TIROPANIS、WENDY HALL、JON CROWCROFT、NOSHIR CONTRACTOR 及 LEANDROS TASSIULAS

网络科学、万维网科学和互联网科学

对具有网络特征的模式的观察，包括生物学、技术和社会学模式，以及对万维网和互联网对社会和商业的影响的观察，都促使我们开展跨学科研究，以期加深对这些系统的认识。多年来，针对它们的研究始终是网络科学研究的主题。然而我们注意到，近来涌现了两个新兴的跨学科领域：万维网科学和互联网科学。

网络科学的数学起源可追溯至 18 世纪莱昂哈德·欧拉 (Leonard Euler) 关于图论[15]的开创性工作，其社会科学起源可追溯至此后两个世纪精神病学家雅可布·莫瑞诺 (Jacob Moreno) 25对创立“计量社会学”所做的尝试。此后不久，心理学家、[2]人类学家[23]和其他社会科学家也借鉴了图论的数学框架，开创了一门名为社会网络的交叉学科。20 世纪末叶，随着人们对探索生物、物理和技术系统中的网络的热情空前高涨，社会网络交叉学科实现了进一步扩张。“网络科学”这个词指的是一个跨学科领域，它依靠物理、数学、计算机科学、生物、经济、社会学来融合未必具有社会属性的网络。[1, 26, 35]对网络的研究包括搭建有助于探明网络起源的解释模型，建立用于预测网络发展趋势的预测模型，以及构建可优化网络输出结果的规范模型。网络科学有一项主要宗旨，即找出适用于各类型网络的通用基础原理和法则，并探索相关模式有时不尽相同的原因。凭借迅猛的长势和巨大的影响，互联网和万维网这两种网络都令一众网络科学家陷入遐想。[13]此外，在线社交网络的出现，和对世界级的大规模在线互动展开研究的可能性，都让网络科学家有望获得对网络发展的宝贵而深刻的认识。[24]

万维网科学[6]是一门近期才出现的交叉学科，它不仅从微小的技

重要见解

- **万维网科学和互联网科学的目标分别是探明万维网和互联网的发展过程，并为关于各自发展前景的论证提供信息。尽管它们的研究社区有重叠之处，但这些目标导致它们的研究途径中优先级有所不同。**
- **网络科学的目标是理解网络的发展过程，无论该网络源自何处，具体包括：作为在人和物之间的信息转化和推送网络的互联网，和作为创作和协作网络的万维网。**
- **鉴于它们在知识上的互补性，我们主张分享和协调这三个跨学科社区正在开发的数据研究基础设施。**

图片由IWONA USAKIEWICZ/ANDRIJ BORYS ASSOCIATES提供

术创新层面（微观层面）研究万维网，还将其作为一种会影响全球社会和商业活动的现象加以研究（宏观层面）；在很大程度上，我们可以将它看作万维网社会机器的理论和实践。社会机器是蒂姆·伯纳斯·李 (Tim Berners-Lee) 于 1999 年提出的概念，指的是由人类开展创造性工作而由机器充当媒介的系统。[3]语义万维网和关联数据技术可提供知识表示和推理方法，并实现对社会机器的进一步支持。[20]

要研究万维网及其影响，我们就要采用一种跨学科的方法，不仅要着眼于技术层面，还要关注到社会、政治和商业层面。建立各层面之间的联系，理解它们的相互影响，研究潜在的基础法则，以及探索如何在人类活动的不同领域内利用这种联系，是万维网科学的主要研究内容。万维网科学依托于包括社会科学、计算机科学和工程学在内的多门学科，其中社会科学有人类学、传播学、经济学、法学、哲学、政治学、心理学和社会学。万维网科学研究日程上的一项重点工作是，理解万维网作为一种社会技术现象是如何发展的，我们又如何确保它在未来的几年内继续发展并造福社会。

互联网科学。互联网提供了许多人类活动所必须依赖的基础设施。虽然仅仅经过了几十年的发展，但有一点是显而易见的：互联网一旦瘫痪，社会、商业、经济、国防和政府将遭受高度破坏性的影响。大家往往将互联网的成功归结于分布式管理模式、网络平等原则及其开放性。[14]与此同时，隐私性、安全性、开放性和可持续性也引起了广泛关注，并催生了相关研究，因为它们常常处于互联网论争的中心。[11]我们可以将互联网视为一种基础设施，其社会价值必须得到保护。[18]从万维网的发展、P2P 应用，到最近的云技术，再到不久以后的物联网，离开了互联网这一基础设施都是空谈。有人主张，互联网和万维网的基础设施层必须分离才能推动创新。[4]最近的一份研究[7]明确了互联网需要发展的几个原则性方向，包括可用性、包容性、可扩展性、可持续性、开放性、安全性、隐私性和弹性。这就促使我们对互联网科学展开多学科研究，以期探明互联网沿着以上原则性方向发展的心理学、社会学和经济学影响。所以说，互联网科学是一个新兴的跨学科领域，汇集了网络工程、计算、复杂性、安全、可信性、数学、物理、社会学、经济、政治学和法学等领域的科学家。早期的互联网拓扑结构研究是这种研究方法的绝佳应用示例。[16]

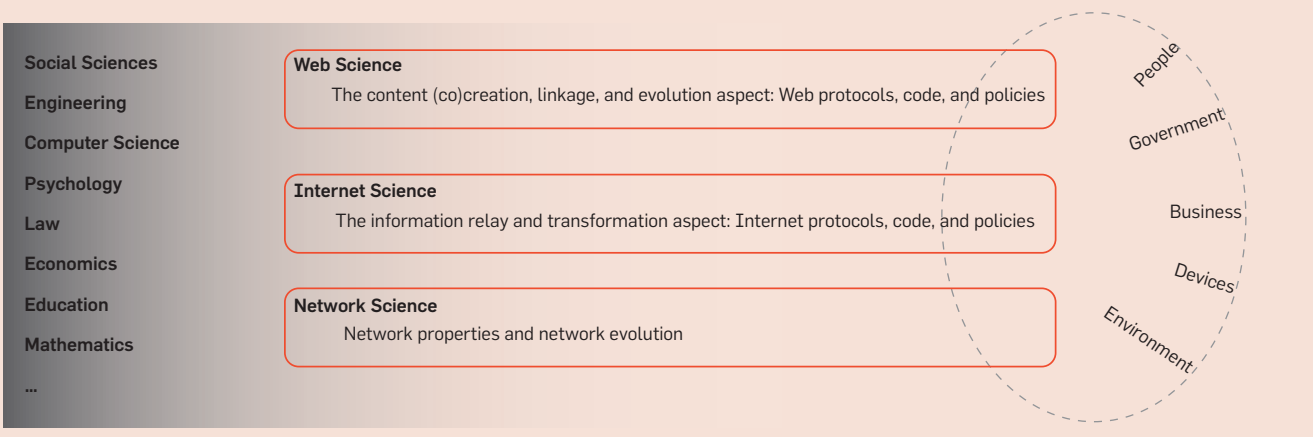
跨学科关系。这三个领域都依托众多学科，分别对万维网、互联网和广义网络的性质，以及它们各自对政府、商业、人民、设备和环境的影响展开研究。然而，各个领域都在研究那些要素如何以独特的方式共同创造和发展，如图 1 所示。万维网科学关注的方面是将那些要素与和它们交互、在它们之间创建联系和阐释它们的内容联系起来。互联网科学关注的方面是在能够引导信息传递和转化的流程中那些要素和资源之间的沟通。而网络科学关注的方面是当我们将这些实体作为网络的组成部分看待时，它们如何展现特定的特征，以及它们又是如何遵循那些有助于理解它们发展方式的基础法则的。

然而，为了更好地理解这些领域之间的相似性和不同之处，并挖掘它们之间潜在的合力，我们需要一个框架进行更详细的对比。

跨学科领域对比

只需要略读这几个跨学科领域[5, 32, 35]的简短描述，我们就会知道，它们在很大程度上都涉及非常相似的学科。用于展现各领域所涉及学科的维恩图显示了这种重叠关系。例如，大家认为心理学和经济学和网络科学、[29]互联网科学、[7]和万维网科学[20]都有关系。这就让我们生出了某些疑问，例如：“如果重叠程度如此之高，是不是可以说这些领域实际上就是同一个领域？”或者

图 1.万维网、互联网和网络科学的各个方面。



“将来它们会互相融合吗？”其他的问题还有：“哪个研究社区和我的研究相关度最高？”或者“各个领域将来会有怎样的发展？”为了探索这些问题，我们提出一个用于深入分析那些跨学科领域的框架，分析的对象包括这些研究社区的形成方式，以及这些共同体在各自的研究中使用的不同话语体系。

研究社区的形成。尽管这三个跨学科领域建立的时间不完全一致，但我们仍可以断言，这些领域内的研究要早于它们正式建立的时间。与此同时，我们还可以说，这几个领域的研究社区形成方式不尽相同。

社会网络社区的形成可追溯至20世纪70年代[17]举办的一系列社会网络大会，包括1975年在达特茅斯(Dartmouth)举办的一次重要会议，这次大会云集了来自美国和欧洲的社会学家、人类学家、社会心理学家和数学家。此后，林·弗里曼(Lin Freeman)于1978年创办了了《*社会网络*》*(Social Networks)*；巴里·威尔曼(Barry Wellman)于1976年成立了国际社会网络分析协会(INSNA)，并于1981年召开了年度阳光地带社会网络大会。自1990年开始，大批物理和生命科学学者也加入社会科学家的队伍，开始探索社会系统中的网络，而且参与者人数日众。2006年召开的年度网络科技(NetSci)大会，2008年美国陆军研究实验室(Army Research Laboratory)为发展跨学科网络科学协作技术联盟(NS-CTA)而投入的大量资金，以及2013年创办的*网络科学期刊*，都印证并推动了这股潮流。[a]显然研究社区已经存在了，并且在那些交叉学科建立之前很早的时候便已从事跨学科工作；我们可以说，网络科学研究社区的形成模式结合了自底向上和自顶向下的方法。

这三个领域都依托众多学科，分别对万维网、互联网和广义网络的性质，以及它们各自对政府、商业、人民、设备和环境的影响展开研究。

2006年，万维网科学研究社区认识到，我们如何看待万维网的影响，对保护它未来的发展至关重要。万维网科学研究中心(WSRI)建立于2006年，后来发展成为国际网络科学协会(WST)，是万维网科学共同体自顶向下形成的一个缩影。WSRI为所有将万维网作为社会技术现象加以深入研究的人擎起了一面旗帜，这些人之中也包括社会网络研究社区。互联网科学社区的形成也遵循了类似模式，其中欧洲网络科学卓越协会(EINS)[b32]是最重要的活动之一，它将该领域的研究社区聚拢在一起。隐私性和网络平等性等领域，已经是互联网科学日程上的重点工作了。

我们可以说，自顶向下的研究社区形成模式能够推动新兴跨学科领域的研究，但为了取得成功，个人、研究机构和行业或政府都必须投入大量资源。尽管万维网科学和互联网科学研究社区基本上是以自顶向下的模式形成的，但是要保证这些研究社区的可持续性，还需要依靠主要研究机构、国家研究委员会和欧盟划拨研究经费，以及个人付出不懈努力。

通用语的使用。除研究社区的形成外，各个领域所使用的话语（即通用语）也有区别。网络科学家最初将图论作为他们的通用语，但近期则借用了从物理过程中提取的模型（渗透、扩散）和博弈论[13]来描述图表中的过程。他们还摒弃了以往的描述性网络指标，转而开发新型的采样推理技术，来检验有关基于多种自组织机制的网络发展方式的假设。[27, 28]因此，图论的使用已经不是当代网络科学研究的必然基础了。此外，还有人用复杂系统分析来处理相变和不同工况之间的不连续；这些方法也用于研究传染病和流行性疾病在全球蔓延的原因。因此，许多网络科学的文章都

a http://journals.cambridge.org/action/displayJournal?jid=NWS

b http://www.internet-science.eu

发表在了像《自然》(Nature) 这样的期刊中。

万维网科学社区自身暂时还没有确立一种通用语，但我们可以说，对万维网标准、技术和模型（HTTP、XML、JavaScript、REST、传播模型、本体论）的理解，以及对社会理论框架的理解，都是有望演化成通用语的组成部分。关于万维网协议及其影响的讨论之中，有一大部分是由 W3C 发起的。对万维网在微观和宏观层面上的发展取得基本认识，是万维网科学研究的基础。

互联网科学社区体采用了类似的话语体系。对于互联网科学而言，根据史蒂文斯 (Stevens) 的著作，[30, 31]通用语的组成部分包括一系列互联网标准 (RFC) 及其相关注释和实施（甚至 C 代码），以及开源领域中实际存在的系统标准实现。其他组成部分还包含对互联网协议原理、基础设施（路由器、链接、AS 级拓扑）、社会科学（偏好依附模型）、法律和政策的基本认识。

研究方法论。在网络科学中，研究方法论牵涉到针对网络进行的网络建模和网络分析，[9, 10]这里所说的网络，包括但绝不限于万维网和互联网。在互联网科学中，最主流的研究方法论涉及测量互联网用户对在线资源和物联网的使用率。万维网科学则广泛采用结合了阐释主义和实证主义方法的混合型研究方法，根据在线社会网络数据集、点击流行为和万维网数据使用来推知万维网的发展过程。

除方法论之外，万维网科学共同体还在尝试打造一个万维网科学天文台，[33, 34]这是一种分布全球的资源，包含与万维网科学相关的数据集和分析工具。与此相似，EINS 项目也在努力为互联网科学研究建立实证基础。而网络科学社区有着悠久的传统，会将标准网络数据集、网络分析软件（如 UCINET[8]）和大型网络数据仓库（如 SNAP[22]）分享给整个社区使用。

显然，这三个领域的研究方法论有一些共同点：

- 它们都依赖从社会网络、基础设施、传感器和物联网搜集而来的数据；
- 它们都涉及测量、建模、模拟、可视化、假设检验、阐释性和探索性研究；并且
- 它们都利用分析技术来量化网络（不论是抽象、虚拟还是真实的）的各项属性，也都采用更多的定性方法。

迄今为止，万维网科学对社会科学、网络科学对社会科学理论和方法论、互联网科学对协议和计算机科学都给予了突出的强调。然而，将来焦点会转移，这三个领域的研究方法或数据也将进一步融合。例如，物联网上的数据可能不再为互联网科学所独用，因为这样的数据可以从万维网科学的视角与万维网上的人类行为数据相结合，或者从网络科学的观点出发来探索它们造就的网络的出现和结果。与此类似，万维网用户行为数据将被用于探索互联网底层基础设施的带宽使用。不同的测量类型指向往往与特定目标相关的事实，这些事实是研究的一部分，对互联网科学和万维网科学这样自顶向下形成的领域而言尤其如此。

尽管方法和数据来源有共通之处，但各个领域都会根据自身的研究途径、以不同的方式混合这些方法，如图 2 所示。制定那些日程的依据是不同的研究目标。

研究目标。“万维网科学关注的是我们如何改善成效，但网络科学关注的是事物的运作机制；”[36]“改善成效”指的是挖掘万维网的潜能并确保它的可持续性。也有人代表互联网科学和网络科学提出了类似的诉求。尽管对“科学”这个术语的使用与知识的系统组织有关，而与目标没有直接联系，但我们认为在这几个跨学科领域形成的过程中，以及它们的研究途径、科学贡献和影响方面，目标的确发挥了一定作用。对万维网科学而言，对万维网本身的研究至关重要，[21]其重要程度堪比对万维网及其发展的保护。[19]对互联网科学而言，互联网及其服务的发展和可持续性是核心目标；众所周知，关于互联网的论争永远不会消停，而接受这种论争恰恰是确保互联网向前走的必要条件之一。[11]似乎对互联网科学和万维网科学而言都是应用研究较早出现，但这种研究应该从基本研究项目的发展中获得信息。另外，万维网科学和互联网科学都不是技术中立的，它们都依赖特定的协议和标准。还有，我们可以说，甚至实施那些标准的代码都嵌入了各个共同体已经达成某种共识的政策。另一方面，网络科学推崇技术不可知论，与互联网科学和万维网科学重叠不多，因为它探索的是网络结构中的新兴结构模式和流，包括社会网络、生物网络、万维网和互联网。最后，万维网科学和互联网科学都属于工程学科；它们的关注点都是构建更好、更强大、更健壮、更高效和更可靠的系统。尽管由于对大数据集的访问，大家对用于预测网络变化的预测分析法，和用于优化网络以完成特定目标的规范分析法都有了更浓厚的兴趣，但网络科学的主要关注点仍是理解和描述出现过程。本质上，网络科学志在利用基础研究中获得的深刻认知来搭建更好的网络。[12]

比较。尽管就研究社区形成模式、通用术语和目标而言，这几个领域确有不同，但它们采用的很多研究方法是一致的。这就意味着，这几个领域在它们重叠的课题上有潜在的合力，而在鲜有重叠或没有重叠的课题上，各大共同体之间也

可能互通有无。图 3 显示了这几个领域中的这类研究课题：

1. 万维网科学：基于万维网的社交媒体领域是万维网科学研究的一个示例。网络方面不是它仅有的部分，因为社交媒体研究聚焦于人和社交媒体资源之间的联系和交互。

2. 互联网科学：互联网科学研究的重点是对物联网如何影响信息的采集和转化，同样不只包含网络研究。

3. 网络科学：运输网络是一个很好的例子，证明了网络科学研究未必和互联网或万维网科学相关。

4. 万维网科学和互联网科学：网络中立性这个例子要求我们同时理解万维网和互联网技术，而且它主要依靠的不一定是网络科学技术。

5. 互联网科学和网络科学：内容分发网络可能需要一定的网络技术来实现配送预测和优化，同时也要求我们理解互联网协议和人与形成这种需求之间的关系。

6. 网络科学和万维网科学：像 Twitter 这样的社交媒体上的信息扩散是一个例子，它既依赖万维网科学社会技术研究方法，也依赖网络分析方法。

7. 万维网科学、互联网科学和网络科学：对网络信任或 SOPA（禁止网络盗版法案）及其副作用的研究依赖所有网络和支持万维网及互联网协议和代码的技术。

鉴于万维网和互联网仍在发展，以上部分研究课题可能会变化。

结论

我们对网络、万维网和互联网科学进行了比较。我们还提出了一个框架，根据几个跨学科领域的研究社区形成、通用语、研究方法和资源以及研究目标来比较它们。通过分析这几个领域中的出版物合著及共引，探索它们在多大程度上是独特或正在融合的跨学科知识社区，我们可以进一步探明这几个领域之间的关系。随着相关的会议和期刊逐渐成熟，这几个领域之间的相似性和区别可能更加明显，这也会让本文的分析变得更容易理解。

互联网科学和万维网科学都能识别技术，它们各自的通用语包括对支撑互联网和万维网的协议和系统的认识，而网络科学则推崇技术不可知论。有人呼吁保持互联网和万维网基础设施层的相互独立性，以推动创新；[4]因而，互联网科学和万维网科学仍然是两个不同的跨

图 2.网络、互联网和万维网科学的方法论。

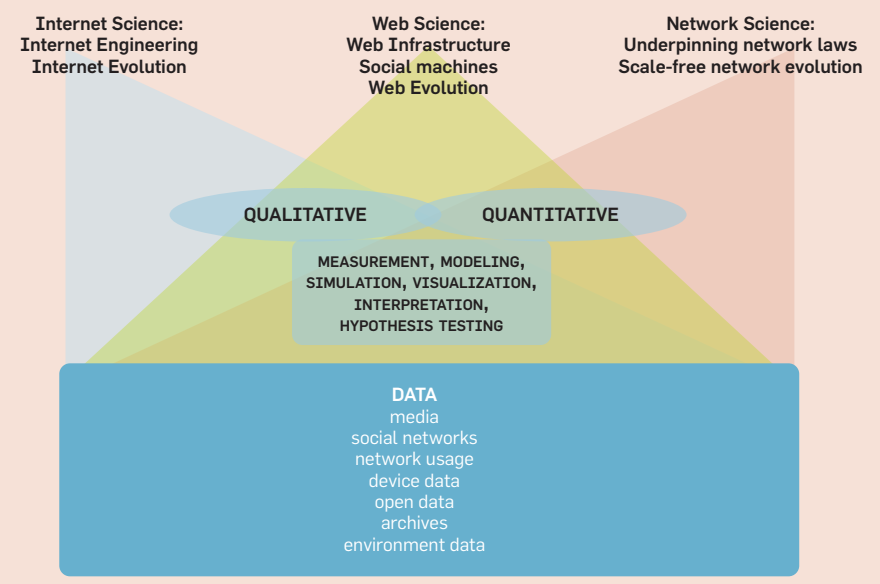
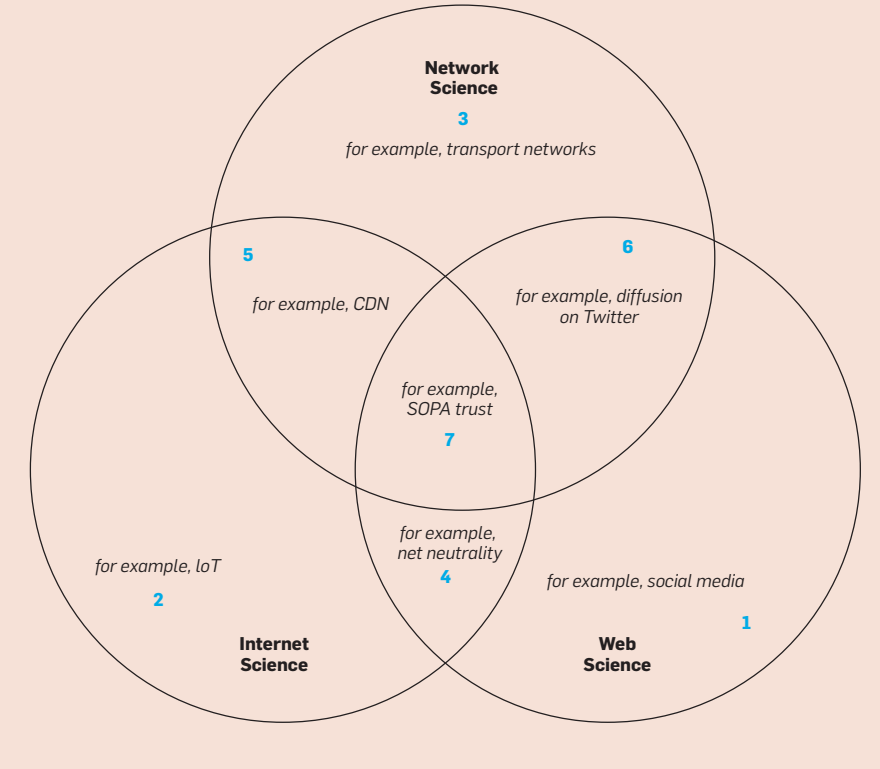


图 3.区分各领域的研究主题和重叠的主题。



学科领域，因为它们有着不同的目标：保护互联网和保护万维网。网络科学探索的对象包括但不限于万维网或互联网。

然而，由于这三个跨学科领域有共通的混合型方法和数据集，通过协作来协调和分享资源将惠及各方；研究人员和投资机构都应该对这项工作予以相当的重视。

鸣谢。 本稿件的撰写获得了来自美国陆军研究实验室 (U.S. Army Research Laboratory) (9500010212/0013//W911NF-09-2-0053)、美国国家科学基金会 (National Science Foundation) (CNS-1010904) 和欧盟 FP7 互联网科学卓越协会 - EINS（拨款协议号 288021）的资金支持。本文包含的观点、意见和/或发现均属于作者，除其他文件特别指出外，不应视为美国陆军部、美国国家科学基金会或欧洲委员会的官方观点、政策或决定。

参考资料

1. Barabási, A.-L. and Albert, R. Emergence of scaling in random networks.*Science 286*, 5439 (1999), 509–512.
2. Bavelas, A. Communication patterns in task-oriented groups.*J. Acoustical Society of America 22*, 6 (1950), 725–730.
3. Berners-Lee, T. *Weaving the Web*. Texere Publishing, 1999.
4. Berners-Lee, T. Long live the Web.*Scientific American*, (2010).
5. Berners-Lee, T., Hall, W., Hendler, J.A., O'Hara, K. and Shadbolt, N. A framework for Web science.*Foundations and Trends in Web Science 1*. 1 (2006b), 1–130.
6. Berners-Lee, T., Hall, W., Hendler, J., Shadbolt, N. and Weitzner, D. Computer science enhanced:Creating a science of the Web.*Science 313*, 5788 (2006a), 769.
7. Blackman, C., Brown, I., Cave, J., Forge, S., Guevara, K., Srivastava, L. and Popper, M.T.W.R.*Towards a Future Internet*, (2010).European Commission DG INFSO Project SMART 2008/0049.
8. Borgatti, S., Everett, M.G. and Freeman, L.C. Computer Software:UCINET 6.*Analytic Technologies*, 2006.
9. Börner, K., Sanyal, S. and Vespignani, A. Network science.B. Cronin, Ed. *Annual Review of Information Science & Technology, 41* (2007), 537–607.
10. Carrington, P. J., Scott, J., and Wasserman, S., eds. *Models and Methods in Social Network Analysis*. Cambridge University Press, 2005.
11. Clark, D.D., Wroclawski, J., Sollins, K.R., and Braden, R. *Tussle in cyberspace:Defining tomorrow's Internet*. Aug. 2002.ACM.
12. Contractor, N.S. and DeChurch, L.A. (in press). Integrating social networks and human social motives to achieve social influence at scale.In *Proceedings of the National Academy of Sciences*.
13. Easley, D. and Kleinberg, J. *Networks Crowds and Markets*.Cambridge University Press, 2010.
14. Economides, N. Net neutrality, non-discrimination and digital distribution of content through the Internet. *ISJLP 4* (2008), 209.
15. Euler, L. Konigsberg Bridge problem *Commentarii academiae scientiarum Petropolitanae 8* (1741), 128–140.
16. Faloutsos, F., Faloutsos, P. and Faloutsos, C. On power-law relationships of the Internet topology.*ACM SIGCOMM CCR.Rev. 29*, 4 (1999).
17. Freeman, L.C. *The Development Of Social Network Analysis*.Booksurge LLC, 2004.
18. Frischmann, B.M.*Infrastructure*. OUP USA, 2012.
19. Hall, W. and Tiropanis, T. Web evolution and Web Science.*Computer Networks 56*, 18 (2012), 3859–3865.
20. Hendler, J. and Berners-Lee, T. From the semantic web to social machines:A research challenge for AI on the World Wide Web.*Artificial Intelligence* (2009), 1–10.
21. Hendler, J., Shadbolt, N., Hall, W., Berners-Lee, T. and Weitzner, D. Web Science: an interdisciplinary approach to understanding the Web.*Commun.ACM 51*, 7 (July 2008).
22. Leskovec, J. Stanford large network dataset collection, 2011; http://snap.stanford.edu/data/index.html
23. Mitchell, J.C. *The Kalela Dance; Aspects of Social Relationships among Urban Africans*.Manchester University Press, 1956.
24. Monge, P.R. and Contractor, N.S.*Theories of Communication Networks*.Oxford University Press, 2003.
25. Moreno, J.L.*Who Shall Survive?Foundations of Sociometry, Group Psychotherapy and Socio-Drama (2nd ed.)*. Beacon House, Oxford, England, 1953.
26. Newman, M.E.J.*Networks:An Introduction*. Oxford University Press, 2010.
27. Robins, G., Snijders, T., Wang, P., Handcock, M., and Pattison, P. Recent developments in exponential random graph (p*) models for social networks. *Social Networks 29*, 2 (2007), 192–215; doi:10.1016/j.socnet.2006.08.003
28. Snijders, T.A.B., Van de Bunt, G.G., and Steglich, C.E.G.Introduction to stochastic actor-based models for network dynamics.*Social Networks 32*, 1 (2010).44–60; doi:10.1016/j.socnet.2009.02.004
29. Steen, M.V. Computer science, informatics, and the networked world.*Internet Computing, IEEE 15*, 3 (2011), 4–6.
30. Stevens, W.R.*TCP/IP Illustrated, Vol. 1:The Protocols*, (1993).
31. Stevens, W.R. and Wright, G.R.*TCP/IP Illustrated, Vol. 2:The Implementation*, (1995).
32. The EINS Consortium.*EINS Factsheet*.The EINS Network of Excellence, EU-FP7 (2012); http://www.internet-science.eu/publication/231.
33. Tiropanis, T., Hall, W., Hendler, J. de Larrinaga, Christian.The Web Observatory:A middle layer for broad data.*Dx.Doi.org 2*, 3 (2014), 129–133; doi:10.1089/big.2014.0035
34. Tiropanis, T., Hall, W., Shadbolt, N., de Roure, D., Contractor, N. and Hendler, J. The Web Science Observatory.*Intelligent Systems, IEEE 28*, 2 (2013), 100–104.
35. Watts, D. The 'new' science of networks.*Annual Review of Sociology*, (2004).
36. Wright, A. Web science meets network science. *Commun.ACM 54*, 5 (May 2011).

Thanassis Tiropanis (t.tiropanis@southampton.ac.uk) 是英国南安普敦大学 (University of Southampton) 电子与计算机科学专业万维网和互联网科学小组的副教授。

Wendy Hall (wh@ecs.soton.ac.uk) 是英国南安普敦大学 (University of Southampton) 的计算机科学教授。她曾出任 ACM 主席。

Jon Crowcroft (jon.crowcroft@cl.cam.ac.uk) 是英国剑桥大学 (University of Cambridge) 计算机实验室的传播系统马可尼教授。

Noshir Contractor (nosh@northwestern.edu) 是伊利诺伊州芝加哥市西北大学 (Northwestern University) 麦考克工程和应用科学学院、传播学院和凯洛格商学院的行为科学 Jane S. & William J. White 教授。

Leandros Tassiulas (leandros.tassiulas@yale.edu) 是康涅狄格州纽黑文市耶鲁大学 (Yale University) 电气工程系的 John C. Malone 教授。

译文责任编辑：陈贵海

DOI:10.1145/2791287

如需获取配套的论文，
请访问doi.acm.org/10.1145/2791285

技术视角
获得群众的能力

Aniket（Niki）Kittur

计算领域早期开拓者（如Herb Simon和Allen Newell）意识到，人类的认知可以用信息处理来构筑。如今，类似于后面的论文所描述的研究正在展示无缝连接人类和机器信息处理器的各种可能性，以便让人们用此前无法想象的方式完成创造性任务。随着网络众包的兴起，这项研究成为了现实。在网络众包中，可以在全世界范围内招募数百万工作者来从事近乎您能想象得到的任何任务。对于某些类型的工作和计算机科学的某些领域，这些条件促成了一种复兴，其中大量的信息被并行化，并得到了众多人类工作者的标记，从而为各领域生成了前所未有的训练集，其范围从自然语言处理一直延伸到计算机视觉。

然而，诸如写作或设计等复杂和创造性的工作却不够简单，很难分解和并行化。例如，设想一下，把由100名工作者组成的群体放到您的下一篇文章上，指示他们修改任何能够改进的地方。您很快便能预见协调大众时所面临的挑战，其中包括避免重复劳动，处理冲突的观点以及创建一个能够容忍缺乏全局观或专长的个人的鲁棒系统。因此，当今众包中普遍存在的并行化独立任务似乎很难满足写作和编辑任务所需的大量互动。在利用群众的能力来处理复杂和创造性的现实世界任务方面，这些协调和互动问题已经成为了重大障碍。

后文作者介绍和实现了一个激动人心的景象，即由众包工作者支持的交互式系统（此处为文字处理器），他们利用系统完成了复杂的认知任务，如明智地缩短文本或作为灵活的“人类宏”。这一景象超越了此前的“绿野仙踪（Wizard of Oz）”式方法（其中人类被用于难以用程序实现的原型功能），它永久地把人类认知带入了交互式系统。这种“群体支撑的系统”能够让人们创造出全新的、现在尚无法实现的计算支撑形式，并积累有助于发展人工智能（AI）的训练数据。

实现这一景象时，核心的挑战是协调群众来完成难以分解的、相互依赖的任务；例如，在一段文章中，某个句子需要与下一句合并。该文作者介绍了一种名为*寻找 - 修改 - 验证*（*Find-Fix-Verify*）的群体编程模式，它对任务进行了分解，让一部分工作者确定需要修改的区域，另一部分工作者修改被最多的人确定的区域，还有一部分工作者选择最佳的修改方式。没有任何一个人需要处理（或者说，甚至阅读）整篇文章，他们的工作专注于若干关键区域，同时也保持了这些区域内部的上下文。该文作者展示了下列证据，使用该模式后，群体能够集体完成任务，产生高质量的输入——包括缩短文本、审校以及遵从开放式的指示——尽管个体的错误率相对较高。

人们可能会问，此类方法如何在时间或复杂度方面实现伸缩。就时间而言，在众包市场中，可能需要承受因等待工作者接受任务而产生的延迟问题。在每种情况下，这确实占用了大部分时间（~20分钟）。就复杂度而言，作者们承认，系统支持的相互依赖程度存在重要的限制；例如，那些需要修改一大片区域的变化或需要修改相关但位置不同的区域的变化有可能引发质量问题。

然而，在加快众包工作的速度、提高其质量和增加其复杂性方面，该领域（包括作者们）已经得到了巨大的进步。在开发了各种方法（如向工作者支付报酬，让他们定期拿取工资）后，招募众包工作者所需的时间从若干分钟降到了若干秒，这样便可支持时间敏感应用（如帮助盲人用户查看周围环境）。由于下列各方面的研究取得的进步，众包工作的质量提高了几个数量级，其范围包括更先进的任务设计【例如，使用贝叶斯真理血清（Bayesian Truth Serum），其中工作者可以预测其他人的回答】，到利用工作者的行为轨迹（例如，观察工作者的做事方式，而不是他们的输出）再到推断工作者在各任务中的质量然后相应地重新调整它们的影响权重。

或许，对于众包工作的未来而言，最重要的问题是它是否能够扩大规模，承担体现人类认知顶峰的、高度复杂和极具创造性的任务（如科学、艺术和创新）。正如作者们、我自己和其他人曾经论证的那样（例如，如*《群体工作的未来》*所述），在支撑群体工作者参与那些我们期望我们的后代能够从事的，有意义的，有影响力的工作类型方面，此项工作可能甚为关键。实现这一前景需要进行高度跨学科的研究，以应对从激励设计到声誉系统，再到管理相互依赖的工作流之中所存在的根本性挑战。由于需要应对朝分散式工作发展的全球趋势所面临的大转变以及伦理问题，此类研究将会相当复杂，但最终的影响也将更为深远。最近已经有若干研究使用群体完成了复杂的创造性工作，包括新闻写作、电影动画、设计批判，甚至是创造新产品，这让人觉得前途充满希望。然而，最好（或最坏）的事情或许尚未来临：我们现在正站在拐点处，如果我们齐心协力，那么计算研究能把我们引向一个由群体支撑的系统所构成的、积极的未来。 C

Aniket (Niki) Kittur是宾夕法尼亚州匹兹堡卡内基梅隆大学人机交互研究所的副教授兼Cooper-Siegel主席。

译文责任编辑：唐杰

DOI:10.1145/2791285

Soylent：内置群众的文字处理器

Michael S. Bernstein，Greg Little，Robert C. Miller，Björn Hartmann，Mark S. Ackerman，David R. Karger，David Crowell，Katrina Panovich

摘要

本文介绍了把众包得到的人类贡献直接融入用户界面时所使用的架构和交互模式。我们的重点放在写作和编辑上面，它们是横跨多个概念和语用活动层级的复杂工作。创作工具能在语用方面提供帮助，但对于更高层级的帮助，撰稿人通常会转而求助他人。因此，我们推出了 Soylent，它是一种文字处理界面，能让撰稿人召集 Mechanical Turk（土耳其机器人）工作者缩短、审校和按照需要以其他方式编辑他们的文档的各个部分。为了提高工作者的质量，我们引入了寻找 - 修改 - 验证（Find-Fix-Verify）的群体编程模式，它把任务分成了一系列生成和评审阶段。评估研究证明了众包编辑的可行性，并调查了编辑的可靠性、成本、等待时间和工作时间等问题。

1. 引言

文字处理是一项复杂的任务，它触及了人机交互方面的很多目标。它支持深度认知活动——写作——并要求复杂的操控。写作相当难：即时是专家也经常会犯风格、语法和拼写错误。那么，当撰稿人做出高层级的决定时，比如把一段文章的时态从过去时改成现在时，或者把草拟的引用充实为真正的引用部分时，她会面临一个艰巨的任务，即在整篇文章的范围内执行巨量的重要任务。最后，当文章可能只有半页纸那么长时，交互式的软件也几乎无法帮助我们精简纸上的短短几段。好的用户界面能够帮助人们完成这些任务；好的人工智能也能提供帮助，但是我们都明白，还有很长的路要走。

在我们的日常生活中，当我们需要获取帮助来完成复杂的认知和操纵任务时，我们通常会求助他人。写作也不例外[5]：我们常常会邀请朋友和同事来帮助我们修改作品，润色文字。但是我们不能总依赖他们：在 10 页长的文章中，同事不愿意审校我们写的每一个句子，或是在每一段里面都删除几行，或是帮助我们按 ACM 风格设置 30 个引用的格式。

Soylent 是一种文字处理界面，它使用了群体的贡献来帮助人们完成复杂的写作任务，它的范围从防止错误和精简段落一直延伸至任务的自动化，如查找引用和改变时态。使用 Soylent 时，感觉就像在写作时有一名编辑全身心地在旁边服务。我们假定，拥有书面英语基本知识的群体工作者能够协助写作新手和写作专家。这些工作者会执行撰稿人可能不愿做的任务，比如仔细地检查需要删减的文本，或是更新地址列表，加入邮编。他们还能解决人工智能尚无法解决的问题，例如，标记出文字处理器无法发现的写作错误。

通过在微软 Word 中整合来自亚马逊 Mechanical Turk 的收费群体工作者，Soylent 可以在写作过程中提供帮助。*Soylent 是人*[a]*:* 它的核心算法涉及招募 Mechanical Turk 工作者（Turker）。Soylent 由三个主要部分组成：

1. *Shortn*，它是一种文本缩短服务，把选中的文本平均精简到原文长度的 85%，同时不改变文本的意义或引入写作错误。
2. *Crowdproof*，它是由人类支撑的拼写和语法检查器，可以找出 Word 漏掉的问题，解释错误并给出修改建议。
3. *Human Macro*，它是一个界面，用于转移任意的文字处理任务，比如为引用设定格式，或者找到合适的图形。

Soylent 的主要贡献是*在交互式的用户界面中融入收费群体工作者这一理念，让它可以根据需要支撑复杂的认知和操纵任务*。本文阐述了一个此类系统的设计，它是一种内嵌在微软 Word 中的实现，也是一种能够提高收费群体工作者在复杂任务上的可靠性的编程模式。接下来，我们对三个主要部分的性能、成本和时延进行了可行性研究，并讨论了我们的方法在隐私、延时、成本和领域知识方面的局限，拓展了上述阐述。

本系统做出的基础性技术贡献是一种名为*寻找 - 修改 - 验证*（*Find-Fix-Verify*）的群体编程模式。Mechanical Turk 需要耗费金钱，而且容易出错；为了让用户觉得它物有所值，我们必须控制成本，保证正确性。寻找 - 修改 - 验证（Find-Fix-Verify）把复杂的群体智能任务分成一系列的生成阶段和利用独立的一致和表决来获取可靠结果的评审阶段。例如，寻找 - 修改 - 验证（Find-Fix-Verify）并不只要求一位群体工作者来阅读和编辑整个段落，它招募一组工作者来找出需要改进的区域，另一组工作者来对候选区域提出改进方案，再由最后一组工作者来排除不正确的候选区域。这一过程可以防止行为不当的群体工作者做了太多或太少的工作，或者把错误带入文档。

[a] 谨向Charlton Heston（1973）致歉：*Soylent 由人组成*。

本文的完整版本发表在《*ACM UIST 2010*会议论文集》中。

在本文的剩余章节中，我们会介绍 Soylent 及其主要部分：Shortn、Crowdproof 以及 Human Macro。我们详细描述了用于实现 Soylent 的寻找 - 修改 - 验证模式，然后评估了寻找 - 修改 - 验证和我们的三个部件的可行性。

2. 相关研究

Soylent 与两个领域的研究

有关：众包系统以及文字处理中的人工智能。

2.1. 众包

收集数据来训练算法是众包的常见用途之一。例如，ESP Game[19] 收集图像中的对象的描述信息用于对象识别。现在，人们已经使用 Mechanical Turk 来为机器视觉 [18] 和自然语言处理 [17] 收集标记后的数据。Soylent 处理了当前那些 AI 算法无法解决的问题，即便拥有充足的数据也无法解决。不过，Soylent 的输出可以用于训练未来的 AI。

Soylent 基于在应用和服务中内置按需选择劳动力的研究。例如，Amazon Remembers 使用 Mechanical Turk 来找出与用户在手机上所拍摄的图片相匹配的产品，PEST[16] 使用 Mechanical Turk 来审查广告推荐。这些系统由单一的用户操作组成，几乎没有或者完全没有交互。Soylent 把这一研究拓展到了更具创造力的、更复杂的任务上，其中用户可以做出个性化的请求，并与直接操作所返回的数据交互。

Soylent 对人类计算的使用意味着它的行为在很大程度上依赖于众包系统的质量，特别是 Mechanical Turk 的质量。Ross 等人发现，Mechanical Turk 有两大主要群体：受过良好教育，收入中等的美国人以及来自印度的，受过良好教育但较不富裕的年轻工作者。[15] Kittur 和 Chi[8] 考虑过如何在 Mechanical Turk 上开展用户研究，他们提出使用可验证的量化问题作为验证机制。寻找 - 修改 - 验证基于需要使用验证来控制质量这一理念。Heer 和 Bostock[6] 探讨了使用 Mechanical Turk 作为图像感知实验的试验床时的情况，当他们实施类似质量检验的基本措施时，他们得到了可靠的结果。Little 等人 [11] 提倡在 Mechanical Turk 上使用人本计算算法。寻找 - 修改 - 验证可作为人本计算算法的一个新设计模式。它特别适于控制懒惰的和过于勤快的 Turker，确定哪些编辑归属于相同的问题，并在界面上用可视化的方式予以展现。Quinn 和 Bederson[14] 已经撰文论述了对人本计算系统的调查，进一步阐述了本文中的简短综述。

2.2. 文字处理中的人工智能

Soylent 受到撰稿人依赖朋友和同事来修改作品，润色文字这一现象的启发。[5]

在 Mechanical Turk 上，审校正在成为一种普通的任务。Standard Minds[b] 提供了由 Mechanical Turk 支撑的审校服务，它通过网页表单的形式接收纯文本，1 天后返回编辑结果。相比之下，Soylent 内置在文字处理器内，它的延时要低很多，而且它在微软 Word 的用户界面中展现编辑结果。我们的研究还贡献了寻找 - 修改 - 验证模式，用于提高此类审校服务的质量。

自动审校的研究历史相当长 [9]，且在多个文字处理器中得到了成功的应用。然而，微软 Word 的拼写检查器常常受到假正错误的困扰，特别是在处理专用名词和不常见的姓名时。它的语法检查器则受到相反的问题的困扰：它会漏掉明显的严重错误。[c] 人类的检查者现在更为可靠，同时还能针对他们发现的错误给出修改建议，而 Word 有时候却做不到——例如，考虑下微软 Word 通常给出的（但是基本没用的）反馈，“片断；请考虑修改（Fragment; consider revising）。”

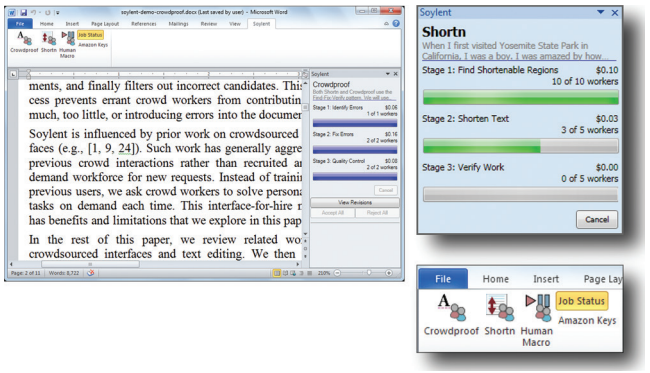
Soylent 的文本缩短组件与文档摘要有关，这一领域也得到了大量研究的关注。[12] 微软的 Word 提供了一个使用句子抽取的文摘功能，它识别出段落中需要保留的多个完整句子，然后删除其他的部分，据此大大缩短了长度，但也牺牲了相当多的内容。Shortn 的方法（它允许重写或删减句子部分）是句子压缩的一个例子。这方面是一个活跃的研究领域，但却受到缺乏训练数据的困扰。[3]Soylent 的结果生成了训练数据，可以帮助推动这个研究领域向前发展。

Human Macro 涉及用于最终用户编程的 AI 技术。若干系统允许用户通过示范重复的编辑任务来实现自动执行；举例而言，如 Eager、TELS 和 Cima。[4]

3.SOYLENT

Soylent 是一个众包文字处理界面的原型。它现在内置在微软的 Word 中（图 1），Word 是一个相当受欢迎的文字处理器，一个提高生产率的应用。它展示了计算系统可以通过联系群体来：（1）创造新的交互支持类型用于文本编辑；（2）扩展人工智能系统，如风格检查，以及（3）支持自然语言指令。这三个目标内置在 Soylent 的三个主要功能中实现：文本缩短、审校和任意的宏任务。

图 1.Soylent把一组群体支持的命令添加到文字处理器中。



[b] http://standardminds.com/.

[c] http://faculty.washington.edu/sandeep/check.

3.1.Shortn：文本缩短

Shortn 旨在展示群体可以支持此前很难实现的新的交互类型和交互式系统。为了让文章符合字数限制，某些作者费尽了心机，在写作过程的最后时刻调整各个段落，去掉若干行文字。这是一项折磨人的工作，而且占用作者时间的理由也不够充分。有的撰稿人写的散文过于冗长，需要人们帮助编辑。自动文摘算法可以识别出文本的相关子集，以便删除。[12] 然而，对于类似 Shortn 中做出的，较小的，局部的语言调整，这些技术的符合程度不高，而且他们无法保证得出流畅的文本。

Soylent 的 Shortn 界面支持作者精简文本的各个部分。用户可以选择非常长的文本区域——例如，一段或一节——然后按下 Word 中 Soylent 命令页签上的 Shortn 按钮（图 1）。作为响应，Soylent 在后台启动了一系列的 Mechanical Turk 任务，并在文本修改好时通知用户。然后，用户可以启动 Shortn 对话框（图 2）。页面左侧是原文段落；右侧是提出的修改。Shortn 提供了一个滑块，支持用户连续地调整段落的长度。当用户执行该操作时，Shortn 会计算出最接近所需长度的，由群体做出的删减的组合，然后把该文本呈现在右侧供用户选择。从用户的角度来看，当她移动滑块来获取更短的段落时，句子会经历轻度编辑、组合乃至完全删除，以匹配长度要求。被编辑或删除的文本局域会以红色高亮的方式可视化显示。在不同的滑块位置，这些区域可能不同。

Shortn 通常会一次删除段落中多达 15%–30% 的部分，经多次迭代后可达 50%。通过鼓励群体工作者专注于赘言，并分开检验重写未改变用户原意，它尽可能地保留了意义。删除整个论点或章节的任务则由用户完成。

3.2.Crowdproof：众包的编辑

Shortn 说明，群体可以支持新的交互类型。我们还可以利用群体来增强应用中内置的人工智能，如审校。Crowdproof 实现了这一理念。

Soylent 提供了一个名为 Crowdproof 的，由人类协助的拼写、语法和风格检查界面（图 3）。这一过程会找出错误，解释问题，并提供一到五个备选的重写文本。Crowdproof 在本质上一个分布式审校器或文字编辑。

使用 Crowdproof 时，用户需要高亮选中一部分文本，然后按下 Soylent 功能区选项卡上的审校（proofreading）按钮。该任务会进入 Soylent 状态面板中的队列，然后用户可以腾出手来继续工作。因为 Crowdproof 需要付费，所以除非接到指令，否则它不会发出请求。

当群体完成工作时，Soylent 会调出错误的部分，在下面加上一条紫色的虚线。如果用户点击错误，会出现一个下拉菜单解释问题，并提供了候选项列表。通过点击期望的候选项，用户可以用他或她的选择来替换不正确的文本。如果用户把鼠标放在错误描述（Error Descriptions）菜单项上方，则会弹出一个菜单，展现其他的第三方意见，用于说明调出这个错误的原因。

3.3. 人类宏（human macro）：自然语言方面的群体脚本录制

在界面上嵌入群体工作者后，我们可以重新思考如何设计短期的最终用户编程任务。通常情况下，用户需要通过脚本语言把自己的意图转换成算法思考，或是明确地通过脚本语言实现，或是通过习得的活动隐含地实现。[4] 但是，传达给人类的任务可以用更自然的方式表述。虽然自然语言命令界面仍然需要继续在相当

图 2.Shortn允许用户通过滑块调整段落的长度。红色的文本指明了群体提供了改写或删减的位置。滑块上的刻度说明了可以获得的长度。

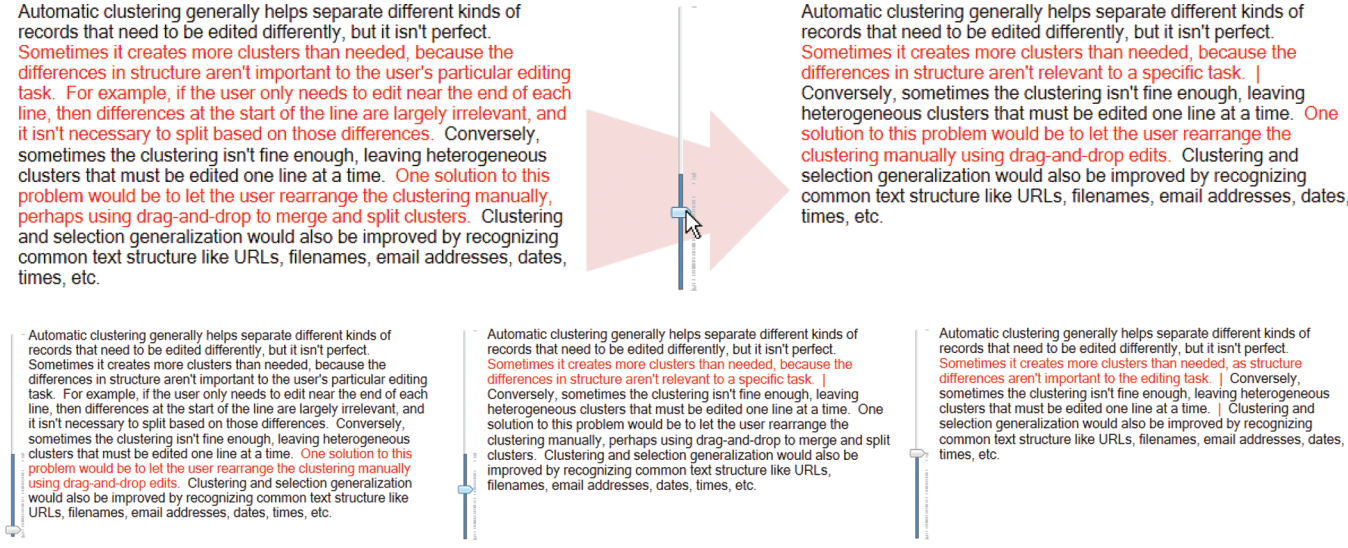
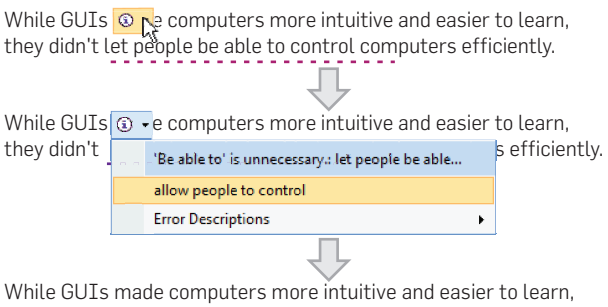


图 3.Crowdproof是一个由人类增强的审校器。下拉菜单解释了问题（蓝色的标题），并给出了修改建议（金色的选区）。



大的搜索空间内处理不受限的输入，但是人类却擅长理解书面指示。

Human Macro 是 Soylent 的自然语言命令界面。Soylent 用户可以通过它用人类语言迅速提出任意工作请求。启动 Human Macro 时，会打开一个请求表单（图 4）。此处主要的设计挑战是确保用户创造的任务会接受妥善的检查，确保其适于 Mechanical Turk 工作者。我们希望防止用户在有问题的命令上消耗费用。

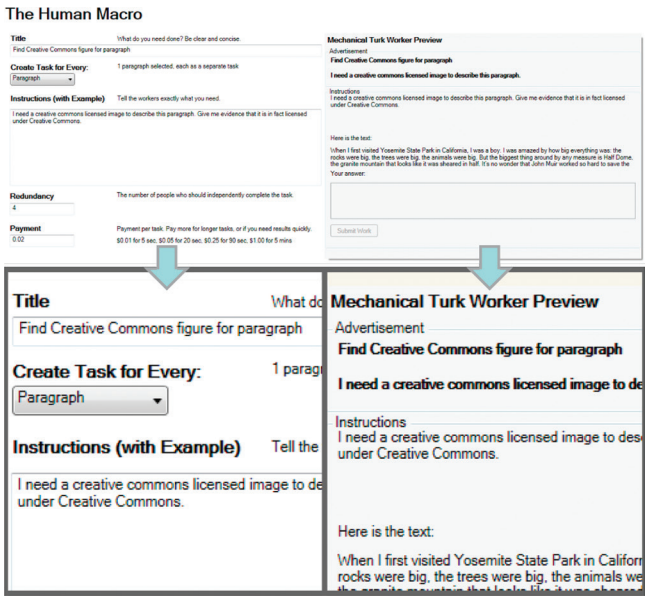
该表单中的对话框分为两个一模一样的部分：左侧是任务输入表单；右侧预览了群体工作者会看到的东西。该预览把用户的请求放在具体的场景中，提醒用户他们正在书写类似 Help Wanted（招募信息）或 Craigslist（克雷格列表）广告的东西。该表单建议用户提供一个样例输入和输出，使用这种方式可以有效地向工作者澄清任务需求。如果用户在打开对话框之前选择了文本，那么他们可以选择按每句或每段来拆分任务，这样一来（例如）处理列表中的所有条目的任务可以并行化。然后，用户选择希望多少位工作者来完成任务。通过支持在某个句子或某段文字上进行一次测试运行，Human Macro 能帮助人们排除任务中的问题。

接下来，用户选择是否应该使用群体工作者的建议来替换现有文本，或仅仅是为现有文本添加注解。如果用户选择替换，则 Human Macro 会在文字下方添加一条紫色的线，然后支持下拉式的替换，这跟 Crowdproof 界面相似。如果用户选择添加注解，则系统会使用 Word 的审核批注接口把反馈填入所选文本上的批注气泡中。

4. 利用群体编程

本节刻画了在利用群体劳动力完成开放式文档编辑任务时面临的挑战。为了处理难以确定群体工作者的工作质量的问题，我们介绍了寻找 - 修改 - 验证模式来提高输出的质量。在我们准备 Soylent，探索 Mechanical Turk 平台时，我们执行并记录了数十次实验。[d] 仅仅在这个项目中，我们就已经利用了 2500 多项不同的任务与 10，000 多名工作者进行了互动。在本文各节中，我们介绍了其中的经验。

图4.Human Macro支持用户对他们的文档提出任意的任务请求。左侧：用户的请求面板。右侧：群体工作者的任务预览，当用户编辑请求面板时会更新。



4.1. 挑战

我们的主要关注点放在群体工作者直接以开放式的方式编辑用户数据的任务上面。这些任务包括缩短、审校和用户发起的改动，如设置地址格式。在我们的实验中，工作者完成任务时得出的很多原始结果并不能让人满意，这点相当明显。凭经验判断，开放式任务中有约 30% 的结果质量不高。这一“30% 法则”也得到了本文实验部分的支撑。很明显，对最终用户而言，30% 的错误率不可接受。为了处理这一问题，我们需要理解不能让人满意的回复的本质，这点相当重要。

工作量方面的高方差。 在投入到任务的工作量方面，群体工作者呈现了相当高的方差。在工作量范围的两端，我们可以刻画两个有用的角色，*Lazy Turker*（*懒惰的 Turker*）和 *Eager Beaver*（*勤快的水獭*）。Lazy Turker 会尽量只做获取报酬需要做的事情。例如，我们要求工作者审校下列从高中论说文网站上获取的，充满错误的段落。[e] 下文用颜色标出了无可争辩的错误，突出了作品中的某些低质量元素：

> The theme of loneliness features throughout many scenes in Of Mice and Men and is often the dominant theme of sections during this story.（孤独的主题是贯穿

[d] http://groups.csail.mit.edu/uid/deneme/.
[e] http://www.essay.org/school/english/ofmiceandmen.txt.

《人鼠之间（Of Mice and Men）》中很多场景的特征。在这一故事中，它往往是各章节中占主导地位的主题。）This theme occurs during many circumstances but is not present from start to finish.（这一主题出现在很多场景中，但不是从头到尾都有。）In my mind for a theme to be pervasive is must be present during every element of the story.（我认为，一个统率全文的主题必须在故事的每一个部分中存在。）There are many themes that are present most of the way through such as sacrifice, friendship and comradeship.（有很多主题在大多数段落中都存在，比如献身、友谊和同志情谊。）But in my opinion there is only one theme that is present from beginning to end, this theme is pursuit of dreams.（但是，我认为，只有一个主题从头到尾一直存在，它就是追求梦想。）

不过，一位 Lazy Turker 在改正一个拼写错误时只会插入一个字符。下文用突出的方式标出了这一修改：

The theme of loneliness features throughout many scenes in Of Mice and Men and is often the dominant theme of sectionsduring this story.（孤独的主题是贯穿《人鼠之间（Of Mice and Men）》中很多场景的特征。在这一故事中，它往往是各章节中占主导地位的主题。）This theme occurs during many circumstances but is not present from start to finish.（这一主题出现在很多场景中，但不是从头到尾都有。）In my mind for a theme to be pervasive is must be present during every element of the story.（我认为，一个统率全文的主题必须在故事的每一个部分中存在。）There are many themes that are present most of the way through such as sacrifice, friendship and comradeship.（有很多主题在大多数段落中都存在，比如献身、友谊和同志情谊。）But in my opinion there is only one theme that is present from beginning to end, thistheme is pursuit of dreams.（但是，我认为，只有一个主题从头到尾一直存在，它就是追求梦想。）

该工作者改正了词语 *comradeship* 的拼写，但漏过了文本中很多明显的错误。事实上，工作者选择做出上述编辑并不令人吃惊，因为在他们的浏览器中，由于这个词有拼写错误，所以它是本来应该用下划线在段落中标出的唯一的词。因此，第一个挑战是劝阻工作者不要做出这种行为。

与 Lazy Turker 一样，Eager Beaver 也会造成麻烦。为了提供帮助，Eager Beaver 超出了任务要求，但在过程中却为用户创造了更多的事情。例如，当要求他改写一个短语时，Eager Beaver 提供了一大串选择：

The theme of loneliness features throughout many scenes in Of Mice and Men and is often the principal, significant, primary, preeminent, prevailing, foremost, essential, crucial, vital, critical theme of sections during this story.（孤独的主题是贯穿《人鼠之间（Of Mice and Men）》中很多场景的特征。在这一故事中，它往往是各章节中占主要、重要、首要、显著、普遍、最重要、必须、关键、必不可少、极其重要的地位的主题。）

该工作者充满热情地把得出的语句变得不合语法。Eager Beaver 还可能在文档中留下额外的批注，或者重新设定段落的格式。把这些结果汇集之后返回给用户可能会有问题。

Lazy Turker 和 Eager Beaver 这两种人都在寻找一种方式来清楚地告诉请求者他们已经完成了工作。在缺乏清晰指导的情况下，Lazy Turker 会选择任何可以生成标识的途径，而 Eager Beaver 则会生成太多的标识。

群体工作者引入了错误。 试图完成复杂任务的群体工作者可能会偶然地引入严重的新错误。例如，当审校评论小说《人鼠之间（*Of Mice and Men*）》的段落时，工作者以各种方式进行了修改，或是把标题删减成《鼠（*Of Mice*）》，或者用他们自己的新错误替代了已有的语法错误，或是改变了文字，声称《人鼠之间（*Of Mice and Men*）》是一部电影，而不是一本小说。如果某位工作者的输出被用作其他工作者的输入，那么此类问题会更加复杂。

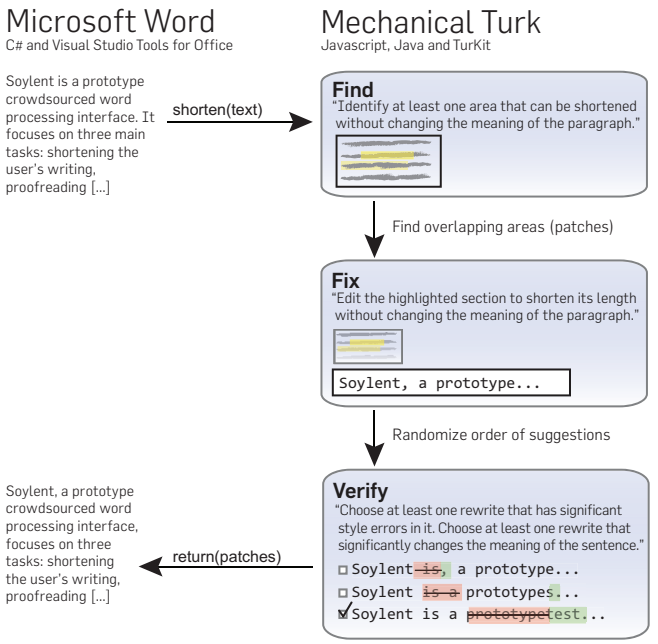
结果：低质量工作。 这些问题综合构成了我们此前所述的 30% 法则：我们从 Mechanical Turk 的工作者处得到的约三分之一的建议没有达到足够高的质量，不能展示给最终用户。我们不能简单地要求工作者来帮助用户缩短或审校一个段落：我们需要引导和协调他们的活动。

这两种角色并不是 Mechanical Turk 所特有的。无论我们使用的是内在的或外在的激励因素——金钱、爱、名誉或其他——参与的分布几乎总是不均匀的。例如，在维基百科中，有很多努力进行编辑工作的

[f] 封闭式任务（如表决）可用于测试有标记的例子，以实现质量控制。[10]开放式的任务可能存在很多正确答案，所以非常重要的标准表决机制不是很有用。

[g] Word的语法检查器，Soylent原文的八名作者和六名检查者均没有发现本句中的错误。后面，Crowdproof发现了错误，并提出了正确的建议，"introduce"应为"introducing"。

图 5.寻找-修改-验证找出了需要编辑的文字块，招募群体工作者来修改文字块，并要求他们进行表决，批准所完成的工作。



Eager Beaver 编辑，但是他们同时也引入了错误，这些错误往往会导致他们的工作被退回。

4.2. 寻找 - 修改 - 验证模式

群体支撑的系统必须控制 Eager Beaver 和 Lazy Turker 的工作，并限制他们引入的错误。对于开放式的任务，如果缺乏合适的控制技术，有问题的编辑所占的比例就会太高，导致无法使用。我们觉得，编程群体的状态与设计模式（如模型 - 视图 - 控制器）引入前 UI 技术的状态类似，设计模式把最佳实践变成了规则。在本节中，我们提出了寻找 - 修改 - 验证模式作为编程群体可靠地完成直接编辑用户数据的开放式任务的一种方法。[f] 我们描述了这种模式，然后阐释了处理各种任务（如审校和文本缩短）时它在 Soylent 中的使用情况。

寻找 - 修改 - 验证的描述。寻找 - 修改 - 验证模式把开放式任务分成了三个阶段，在每个阶段中工作者均可做出明晰的贡献。图 5 用可视化的方式描绘了这一工作流。

Shortn 和 Crowdproof 均使用了寻找 - 修改 - 验证模式。在本节中，我们使用 Shortn 作为示例来说明这一模式。为了给用户提供对段落长度的近乎连续的控制，Shortn 需要生成很多备选的改写方案，同时又不改变原文的意义或引入 [g] 语法错误。首先，我们把输入区域分为多个段落。

第一个阶段（Find）要求几位群体工作者在用户的作品中找出需要加以注意的文字块。例如，进行缩短时，Find 阶段会要求十名工作者找出至少一个需要缩短的短语或句子。任何一个单独的工作者都可能会产出充满噪声的结果（例如，Lazy Turker 可能倾向于找出靠近段落开头的错误）。Find 阶段整合了各种独立的观点，以找出工作者认同程度最高的问题：多人独立得出的一致结果通常是一种说明群体正确性的强信号。Soylent 保留了至少有 20% 的工作者认为需要修改的文字块。然后，这些结果被并行地送入 Fix 阶段。

Fix 阶段会招募工作者来修改工作者一致认为需要注意的每一个文字块。现在，每一个任务均由对感兴趣区域的受限编辑组成。工作者会看到在段落中突出显示的文字块，然后我们会要求他们改正问题（例如，缩短文本）。工作者能看到整个段落，但是只能编辑包含那个文字块的句子。少数的几位（3-5）工作者提出修改建议。即使 30% 的建议是不好的，3–5 名份建议也足以生成切实可行的备选方案。在 Shortn 中，工作者还可以就文字块是否可被完全删除进行表决。如果可以的话，我们引入了空串作为修改方案。

Verify 阶段对修改进行了质量控制。我们随机安排了在 Fix 阶段中生成的，独一无二的备选方案的顺序，然后要求 3-5 名新的工作者对它们进行投票（图 5）。我们或是要求工作者投票选出最佳的选项（当界面需要一个默认选择时，如 Crowdproof），或是要求他们标出不好的建议（当界面要求尽可能多的选项时，如 Shortn）。为了确保工作者不会对自己的工作投票，我们禁止所有参与过 Fix 的工作者参与该段文字的 Verify 阶段。为了便于比较，Mechanical Turk 任务使用颜色和删除线标注了每一处改写，突出显示了它与原文的区别。我们使用了多数票来删除有问题的改写和确定是否可以删除文字块。在 Verify 阶段结束时，我们拥有了候选文字块的结合，并为每个文字块准备了经验证的改写文本清单。

为了保证该算法的响应速度，我们在每个阶段均使用了 15 分钟的超时设置。如果某个阶段超时，在 Find 中我们仍然会等待至少六名工作者，在 Fix 中等待至少三名工作者，在 Verify 中等待至少三名工作者。

模式的探讨。为什么我们应该把任务分成独立的寻找 - 修改 - 验证阶段呢？为什么不让群体工作者找出错误并改正错误，从而得到更高的效率，节约更多的成本？ Lazy Turker 一直会选择最简单的错误来修改，所以合并 Find 和 Fix 会导致覆盖范围变窄。通过把 Find 与 Fix 分开，我们可以指示 Lazy Turker 对他们可能在其他情况下会忽视的文字块提出解决办法。另外，把 Find 与 Fix 分开还能让我们用并行的方式合并已完成的工作。如果每位工作者均编辑了整个段落，我们将无法知道哪些编辑设法改正的是同一个问题。通过把 Find 和 Fix 分开，我们可以在编辑与文字块之间进行映射，生成更为丰富的用户界面——例如，Crowdproof 替换下拉菜单中的多个选项。

Verify 阶段减少了返回结果中的噪声。此处的高阶理念是，我们让工作者处于彼此间存在积极的紧张关系中：一组工作者正在提出解决办法，而另一组的任务是批判性地看待这些建议。根据经验来说，与书写原文相比，工作者更擅长评论建议。Verify 工作者之间独立达成的一致可以帮助我们证明某个编辑是好

的，或是坏的。验证提高了质量，但用了更长的时间：对于可以忍受更多错误但要求时间间隔短的用户，他可以选择不验证工作，或使用更少的验证工作者。

寻找 - 修改 - 验证也有缺陷。寻找 - 修改 - 验证模式与其他 Mechanical Turk 算法共有的一个挑战是，当工作者接受任务的速度迟缓时，它有可能会停滞。系统并不是等待十名工作者完成 Find 任务之后才进入 Fix 阶段，如果数量达到最低阈值的工作者已经完成了工作，那么超时参数可以强行让我们的算法前进。寻找 - 修改 - 验证还会让特别熟练的工作者很难做出大的改变：把任务分解之后，中等熟练的工作者更容易完成，但是，对于群体中的专家而言，则有可能更让他们感觉沮丧。

5. 实现

Soylent 由前端应用级微软 Word 插件和运行 Mechanical Turk 任务的后端服务组成（图 5）。微软 Word 插件使用微软 Visual Studio Tools for Office（VSTO）和 Windows Presentation Foundation（WPF）编写。后台的脚本使用了 TurKit Mechanical Turk 工具包。[11]

具体来说，给定滑块上的候选长度后，Shortn 必须选择一组重写。当用户确定所需的最大长度后，Shortn 会按长度限制搜索最长的重写组合。简单的实现是穷举所有的组合，然后把它们放入缓存，但在拥有很多文字块时，该方法的伸缩性能相当差。如果运行时间是一个问题，那么我们可以把这种搜索看成多选择背包问题。在多选择背包问题中，你放入背包的项目来自多个类别，但在每个类别中只能选择一个项目。因此，在 Shortn 中，每个项目类别是带有一个或多个选项的文本区域：如果群体没把某个类别选成文字块，则这个类别只有一个选项；如果群体把文本圈出来作为文字块，并写出了备选方案，那么这个类别就有多个选项。多选择背包问题可通过多项式时间的动态规划算法求解。

6. 评估

我们最初的评估旨在为 Soylent 的端到端可行性确定证据以及理解寻找 - 修改 - 验证设计模式的各种性质。

6.1. 对 Shortn 的评估

通过在示例文本上运行 Shortn，我们对 Shortn 进行了定量评估。我们的目标是观察 Shortn 能够把文本缩短多少及其相关的成本和时间特性。我们收集了五份可发送给 Shortn 的示例文本，每一份文本的长度为一到七段不等。我们选择的这些输入的范围从论说文初稿到终稿，从容易理解的材料到技术性强的材料（表 1）。

表1.在评估时，Shortn运行一次后得到的修订文本的长度在原段落长度的78%到90%之间。

输入文本	原长度	最终长度	工作的统计数据	每段耗时（分钟）	样例输出
博客	3段，12句，272个词	83%的字符长度	$4.57美元，158名工作者	46–57	Print publishers are in a tizzy over Apple’s new iPad because they hope to ~~finally~~ be able to charge for their digital editions.But in order to get people to pay for their magazine and newspaper apps, they ~~are going to~~ have to offer something different that readers cannot get at the newsstand or on the open Web.
经典的UIST论文[7]	7段，22句，478个词	87%	$7.45美元，264名工作者	49–84	The metaDESK effort is part of the larger Tangible Bits project.~~The Tangible Bits vision paper,~~ which introduced the metaDESK ~~along with~~ and two companion platforms, the transBOARD and ambient ROOM.
UIST论文初稿	5段，23句，652个词	90%	$7.47美元，284名工作者	52–72	~~In this paper we argue that~~ it is possible and desirable to combine the easy input affordances of text with the powerful retrieval and visualization capabilities of graphical applications.We present WenSo, ~~a tool that~~ which uses lightweight text input to capture richly structured information for later retrieval and navigation in a graphical environment.
杂乱无章的Enron电邮	6段，24句，406个词	78%	$9.72美元，362名工作者	44–52	~~A previous board member,~~ Steve Burleigh~~,~~ created our web site last year and gave me alot of ideas. ~~For this year,~~ I found a web site called eTeamZ that hosts web sites for sports groups.Check out our new page: [···]
技术写作[1]	3段，13句，291个词	82%	$4.84美元，188名工作者	132–489	Figure 3 shows the pseudocode that implements this design for Lookup.FAWN-DS extracts two fields from the 160-bit key: ~~the i low order bits of the key~~ (the index bits) and the next 15 low order bits ~~(the key fragment)~~.

The Example Output column contains example edits from each input.

为了模拟现实世界的部署情况，在运行算法时，我们启用了超时，为每个阶段设置了 20 分钟的超时时间。我们需要 6-10 名工作者来完成 Find 任务，3-5 名工作者完成 Fix 和 Verify 任务：如果 Find 任务连六名工作者都无法招募到，那么它可能会无限地等待下去。为了匹配 Mechanical Turk 的市场价格，我们为每次 Find 付费 $0.08，每次 Fix 付费 $0.05，每次 Verify 付费 $0.04。

每个由此得出的段落都有很多可能的变化，具体取决于通过 Verify 阶段的，已被缩短的备选方案的数量——我们选择了最短的可能版本进行分析，并把它的长度与原文长度进行了对比。我们还测量了等*待时间*（*wait time*），任务发布和工作者接受任务之间的时间以及*工作时间*（*work time*），即接受任务和提交结果之间的时间。在所有的任务中，算法有可能在等待工作者时停滞，这对平均数的影响相当大。因此，我们报告了中位数，它们对异常值更为鲁棒。

结果 Shortn 生成了约为原文长度 78%-90% 的修订版本。作为参考，缩减到 85% 可以把一份 $11\frac{3}{4}$ 页 ACM 论文初稿减少到 10 页，其内容不会出现明显地删减。表 1 汇总并给出了 Shortn 运行情况的示例。通常来说，Shortn 的重点放在不必要的冗长短语上，如“are going to have to”（表 1 中的 Blog）。当文字块超过句子的边界时，群体工作者会合并句子（表 1，经典的 UIST 论文），而且偶尔会删除整个短语或句子。

为了调查时间特性，我们把等待时间的概念从工作时间中分离了出来。Shortn 的绝大多数运行时间现在花在等待上，因为工作者可能需要数分钟或数小时来找到和接受任务。此处，我们当前的等待时间——Find 的中位数、Fix 的中位数和 Verify 的中位数之和——为 18.5 分钟（第一个四分位数 $Q_1 = 8.3$ 分钟，第三个四分位数 $Q_3 = 41.6$ 分钟）。如表 1 所示，因为任务可能在等待工作者时停滞，该等待时间可能会更长。

如果我们只考虑工作时间，且假设等待时间可以忽略，那么 Shortn 可以在数分钟内完成删减。通过检查工作者在寻找 - 修改 - 验证流程中每个阶段所耗时间的中位数，我们再次估算了整体的工作时间。该流程显示，统计所有三个阶段所花费的时间之和时，缩短任务的中位数为 118 秒的工作时间，即略低于 2 分钟，（$Q_1 = 60$ 秒，$Q_3 = 3.6$ 分钟）。使用自本研究发布时起开发的招募技术后，用户可以看到，缩短任务接近 2 分钟的限值。

在我们的付费模型中，缩短段落的平均成本为 $1.41 美元。该成本可分为使用 $0.55 美元来找出平均两个文字块，然后使用 $0.48 美元来生成备选方案，再用 $0.38 美元为其中的每个文字块筛选结果。我们的经验是，付费更少时会减慢流程中靠后的部分，但它不会影响质量 [13]——如果缩短段落的截止日期比较宽松，那么这个方式会是切实可行的。

从定性的角度来说，当输入包含不必要的文本时，Shortn 最为成功。例如，处理 Blog 输入时，Shortn 能够删除若干词语和短语，同时不改变句子的意思。工作者能够轻松地把删除后的文本组织成句子。即便技术性最强的输入文本也有无关的短语，所以在每一段中，Shortn 通常能够做出至少一次此类性质的编辑。当 Soylent 运行时，它能够收集这些直接改写，把它们放入大型数据库，然后使用它们来训练机器学习算法，以自动提供某些缩短建议。

Shortn 偶尔会把错误带入段落中。虽然工作者倾向于不去删除他们无法理解的材料，但是他们偶尔仍会标出这种文字块。结果，工作者有时候会做出一些语法正确但风格上不正确的编辑。例如，从引言中删除带有学术标识的短语“In this paper we argue that…（在本文中，我们论证了 ……）”可能不合适。删减是第二个错误源：Fix 阶段中的工作者会投票表决是否把整个文字块从句子中删除，但是他们没有机会来减小删减句子某部分所造成的影响。所以，删减通常会导致在句子边界处出现大小写和标点问题。现代的自动纠错技术能够发现很多这种错误。并行则是另一个错误源：例如，在技术写作中（表 1），两处删减来自两个不同的文字块，因此会被不同的工作者处理。这些工作者无法预测到他们的删减不相配，因为一名工作者删除了插入成分，而另一名工作者则删除了主短语。

为了调查这些问题的严重程度，我们对所有 126 条缩短建议进行了编码，检查它们是否引起了语法错误。在这些建议中，有 37 条建议是不合语法的，这再次支撑了我们的经验法则，即工作者做出的 30% 的原始编辑充满噪声。在 Verify 步骤中，工作者找到了 19 个错误（37 个中的 50%），同时也删除了 15 个符合语法的句子。因此，它的错误率为（18 个假负错误 + 15 个假正错误）/137 = 26.1%，再次接近 30%。微软 Word 的语法检查器找到了 13 个此类错误。结合 Word 和 Shortn 后，找出了 37 个错误中的 24 个。

我们进行了实验，把 Blog 文本的最短输出再次送回到算法中，检查它能够继续缩短。在每次迭代中，它继续把段落缩短了 70%-80%。在 3 次迭代后，我们停止了算法，结果把文本缩短到了不到 50% 的长度，同时也没有牺牲很多的可读性或丢失很多主要内容。

6.2. 对 Crowdproof 的评估

为了评估 Crowdproof，我们拿到了需要审校的五个输入文本集。这些输入是一些虽然通过了 Word 语法检查器的检查但充满错误的文本，其中有 ESL 学生书写的文章，演示过程中的速记文字，写得很差的维基百科文章（Dandu Monara）以及 UIST 论文草稿。对于这五篇文章，我们手工标出了每篇文章的拼写、语法和风格错误，找出了总共 49 个错误。然后，我们把各阶段超时设置为 20 分钟，再运行 Crowdproof 来处理这些输入，价格设置为 Find$0.06 美元，Fix$0.08 美元，Verify$0.04 美元。我们度量了 Crowdproof 发现的错误，Crowdproof 纠正的错误以及 Word 发现的错误。我们规定，如果某个被工作者识别的文字块中含有错误，则说明 Crowdproof 找到了一个错误。

结果 Soylent 的审校算法发现了 49 个错误中的 33 个（67%）。相比之下，微软的 Word 语法检查器发现了 15 个错误（30%）。综合来看，Word 和 Soylent 标出了 40 个错误（82%）。Word 和 Soylent 两者并非都专注于简单和明显的错误，它们倾向于识别不同的错误。这一结果为 Crowdproof 方法提供了更多的支撑：它能专注于自动的审校器尚无法识别的错误。

在纠正自己发现的错误方面，Crowdproof 相当有效。使用 Verify 阶段来选择最佳的替换文本后，Crowdproof 纠正了它标出的 33 个错误中的 29 个（88%）。为了调查 Verify 阶段的影响，我们给工作者建议的每一种独一无二的改正进行了标注，标明是否是语法方面的。在 62 个建议中，有 28 个（或者说 45%）完全是不合语法的。虽然此类建议充满噪声，但也生成了正确的替换文本，这一事实再次说明，相比创作而言，工作者更擅长审核。

Crowdproof 最常见的问题是，对于包含了更严重的错误的文字块，Crowdproof 漏掉了该文字块中的一个较小的错误。对于含有 Crowdproof 未能纠正的四个错误的文字块，它们至少至少还有一个其他的错误；Lazy Turker 只纠正了最明显的问题。第二个问题是缺乏领域知识：在 ESL 输入中，工作者不知道 GUI 是什么，所以他们也不可能知道作者想用“GUIs”，而不是“GUI”。还有一些风格上面的修改原作者可能不会赞同：在 UIST 草稿的输入中，作者爱好三长划，但工作者并不欣赏这点。

Crowdproof 的运行时间特性有很多与 Shortn 相同。它的工作时间的中位数是 2.8 分钟（Q_1 = 1.7 分钟，Q_3 = 4.7 分钟），所以在相当短的工作时间内就能完成。与 Shortn 相似，它的等待时间为 18 分钟（(Median = 17.6，Q_1 = 9.8，Q_3 = 30.8）。每段文章所耗费的金钱更多（μ = \$3.40，σ = \$2.13），因为它在每段文章中识别出了多得多的文字块：我们选择了急需审校的段落。

6.3. 对人类宏（Human macro）的评估

我们的兴趣是了解最终用户是否能够指导 Mechanical Turk 工作者来执行开放式任务。用户能否清晰地传递他们的意图？工作者能够正确地执行业余者编写的任务？

方法。我们生成了五个可行的 Human Macro 场景。这些场景包括把故事的时态从过去时改成现在时，找出拥有 Creative Commons（知识共享）许可的图像来说明一段文字，为博客帖子草稿提供反馈，收集 BibTeX 用于生成某些引用，以及填写列表中的邮寄地址。我们招募了两组用户：我们计算机科学系的五名本科生和研究生（4 名男性）以及我们系的五名行政助理（均为女性）。我们从五条提示信息中挑出一条信息向每位用户展示，其由样例输入和输出对组成。我们故意不向参与者描述任务，这样我们就不会影响他们书写任务描述的方式。然后，我们向参与者介绍了 Human Macro，并描述了它的用途。然后，我们要求他们使用 Human Macro 来为他们的提示信息书写一篇任务描述。然后，我们把描述发给了 Mechanical Turk，要求五名工作者完成每个请求。除了我们的参与者生成的十个请求外，一名作者自己生成了五个请求，用以仿真熟悉 Mechanical Turk 的用户。

我们使用了两个质量度量来衡量结果：意图（工作者是否理解了提示信息，且做出了善意的努力？）以及准确度（结果是否没有缺陷？）。如果工作者完成了工作，但犯了小错误，那么结果可视为意图良好，准确度不佳。

结果通常情况下，用户能够成功地传递他们的意图。在意图方面，命令平均达到了 88% 的成功率（max = 100%，min = 60%）。当提示信息包含两个要求时，会发生典型的意图错误：例如，Figure 任务要求找出一张图片并证明图片是得到 Creative Commons（知识共享）许可的。工作者读到理解需要他们找出一张图片时就再往下读了，他们找到了一张图片，然后离开了。成功的用户给他们的请求的标题字段中清楚地标出了 Creative Commons 状态。

在准确度方面，我们再次看到约 30% 的工作含有错误。（平均准确度为 70.8%。）通常，工作者完成的任务大体上是正确的，但在某些细节上犯了错误。例如，在 Tense 任务中，除了一个动词之外，某些工作者把其他所有的动词改成了现在时态，在 List Processing 任务中，有时候某个字段的大小写不正确，或者某位 Eager Beaver 添加了太多的其他信息。把含有此类错误的结果交给用户是相当危险的，因为用户可能同样不会意识到工作中有小错误。

7. 讨论

本节回顾了 Soylent 所体现的，与收费的群体工作界面的本质有关的一些基本问题。我们的工作说明，或许可以实现一种转变，从把 Wizard of Oz 技术仅仅作为原型工具的时代转变到“Wizard of Turk”可以永久地嵌入系统的时代。我们探讨了等待时间、成本、法律上的所有权、隐私和领域知识所引发的各种问题。

在我们的界面外包设想中，作者可以立即获取人类专家池。由于工作者的人口特性、工作者的空闲情况以及任务本身的相对吸引力等因素，在我们当前的实现中，延迟时间仍在分钟到小时的级别上。随着众包工作在未来的增长，延迟时间可能会缩短，所以这是未来研究的一个重要领域。或许可以通过更高的金钱投入来明确地提高响应度，或者在任务需要前用其他任务留住工作者 [2]。

在成本方面，Soylent 要求作者为所有完成文档编辑的工作者付费——即使很多修改永远都不会进入最终的作品。因此，人们可能会争辩说，界面外包价格太过昂贵，不切实际。我们反驳道，事实上当前所有的文档处理任务也会产生明显的成本（从计算的基础设施、时间、软件和工资方面来说）；唯一的区别是界面外包精确地量化了每一小部分工作的价格。需要为每次编辑付费可能会把它的用途限制在商业文本上，不过作者是否值得花那么多的成本来提高生产率仍然是一个悬而未决的问题。

在隐私方面，Soylent 让第三方工作者接触了作者的文档，同时还不知道工作者的身份。如果文档的内容是保密的，或者在其他方面是敏感的，那么作者和他们的雇主可能不愿意做出这样的披露。一个解决方案是，限制可以执行任务的工作者集合：例如，大型公司可以维护自己的内部工作者池。隐私和披露并不是一种二元的对立，它们是一种连续体。

在所得文本的法定所有权方面，Soylent 也提出了疑问，因为它部分是由用户生成的，部分是由 Turker 生成的。参与寻找 - 修改 - 验证阶段的 Turker 是否对文档拥有任何法定权利？我们认为他们没有：Mechanical Turk 的工作者合同明确地说明工作成果是职务作品，所以成果属于请求者。历史上也有类似的先例：对于他们对文章所做的编辑，传统的文字编辑并不拥有所有权。然而，众包的界面需要仔细地思考法律方面的问题。

研究社区应该思考如何把众包变成一种社会公益事业，而不是一种加剧不平等的工具，这点相当重要。众包是科学管理的一种复兴。在优化工作流方面，泰勒主义有着积极的影响，但是它也与工厂工作的非人性化因素和工业革新有关联。与此类似，缺乏经验的众包可能会把人当成一种新的抽象 API 调用类型，从而忽视了这些社会技术系统中核心的人性方面。与此相反，我们需要演进众包系统的设计过程，以直接融入群体工作者的看法。

8. 结论。

下列结论被 Shortn 到了 85% 的长度：本文介绍了 Soylent。Soylent 是一个利用群体工作者帮助人们完成审校、文档缩短、编辑和批注任务的文字处理界面。Soylent 是新型的交互式用户界面~~的示例~~，其中最终用户可以直接获取一群工作者的协助，协助他们完成需要人类注意力和常识的任务。实现这些界面~~类型~~需要用于~~界面软件的~~新的软件编程模式，因为大众与计算机系统的行为不同。我们介绍了一个重要的模式，寻找 - 修改 - 验证，它把复杂的编辑任务分成了一系列的识别、生成和验证阶段，这些阶段~~使用独立得出的一致和表决来~~生成可靠的结果。我们用一系列的编辑任务对 Soylent 进行了评估，~~在结合自动检查机制后~~，我们找到和纠正了 82% 的语法错误，每次迭代把原文长度缩短到了约 85%，并成功地执行了各种各种的人类宏。

未来的工作可以分为下列三个类别。首先是用于文字处理的，由群体驱动的新特性，比如可读性分析、智能查找替换~~（以便在把“Michael”替换成“Michelle”的同时，把“he”改成“she”）~~以及图片或引用编号的检查。其次，开发用于优化群体编程算法的新技术，以减少等待时间和成本。最后，我们认为，在把按需的群体工作集成进其他的创作界面方面（特别是在~~如图片编辑和编程等~~创造性领域方面），我们的研究指明了方向。 C

参考资料

1. Andersen, D.G., Franklin, J., Kaminsky, M., Phanishayee, A., Tan, L., Vasudevan, V. FAWN:A fast array of wimpy nodes.In *Proc.SOSP '09* (2009).
2. Bigham, J.P., Jayant, C., Ji, H., Little, G., Miller, A., Miller, R.C., Miller, R., Tatrowicz, A., White, B., White, S., Yeh, T. VizWiz:Nearly real-time answers to visual questions.In *Proc.UIST '10* (2010).
3. Clarke, J., Lapata, M. Models for sentence compression:A comparison across domains, training requirements and evaluation measures.In *Proc.ACL '06* (2006).
4. Cypher, A. *Watch What I Do.*MIT Press, Cambridge, MA, 1993.
5. Dourish, P., Bellotti, V. Awareness and coordination in shared workspaces.In *Proc.CSCW '92* (1992).
6. Heer, J., Bostock, M. Crowdsourcing graphical perception:Using Mechanical Turk to assess visualization design.In *Proc.CHI '10* (2010).
7. Ishii, H., Ullmer, B. Tangible bits:Towards seamless interfaces between people, bits and atoms.In *Proc.UIST '97* (1997).
8. Kittur, A., Chi, E.H., Suh, B. Crowdsourcing user studies with Mechanical Turk.In *Proc.CHI '08* (2008).
9. Kukich, K. Techniques for automatically correcting words in text.*ACM Comput.Surv.(CSUR) 24*, 4 (1992), 377–439.
10. Le, J., Edmonds, A., Hester, V., Biewald, L. Ensuring quality in crowdsourced search relevance evaluation:The effects of training question distribution.In *Proc.SIGIR 2010 Workshop on Crowdsourcing for Search Evaluation* (2010), 21–26.
11. Little, G., Chilton, L., Goldman, M., Miller, R.C. TurKit:Human computation algorithms on Mechanical Turk.In *Proc.UIST '10* (2010) ACM Press.
12. Marcu, D. *The Theory and Practice of Discourse Parsing and Summarization.*MIT Press, Cambridge, MA, 2000.
13. Mason, W., Watts, D.J.Financial incentives and the performance of crowds.In *Proc.HCOMP '09* (2009) ACM Press.
14. Quinn, A.J., Bederson, B.B.Human computation:A survey and taxonomy of a growing field.In *Proc.CHI '11* (2011) ACM.
15. Ross, J., Irani, L., Silberman, M.S., Zaldivar, A., Tomlinson, B. Who are the crowdworkers?Shifting demographics in Amazon Mechanical Turk.In *alt.chi '10* (2010) ACM Press.
16. Sala, M., Partridge, K., Jacobson, L., Begole, J. An exploration into activity-informed physical advertising using PEST.In *Pervasive '07*, volume 4480 of *Lecture Notes in Computer Science* (Berlin, Heidelberg, 2007), Springer, Berlin, Heidelberg.
17. Snow, R., O'Connor, B., Jurafsky, D., Ng, A.Y.Cheap and fast—But is it good?Evaluating non-expert annotations for natural language tasks.In *Proc.ACL '08* (2008).
18. Sorokin, A., Forsyth, D. Utility data annotation with Amazon Mechanical Turk.*Proc.CVPR '08* (2008).
19. von Ahn, L., Dabbish, L. Labeling images with a computer game.In *CHI '04* (2004).

Michael S. Bernstein（msb@cs.stanford.edu）来自加利福利亚州斯坦福市斯坦福大学。

Björn Hartmann（bjoern@cs.berkeley.edu）来自加利福利亚州加利福利亚大学伯克利分校计算机科学学院。

Greg Little（glittle@csail.mit.edu）

Mark S. Ackerman（ackerm@umich.edu）来自密歇根州安娜堡密歇根大学计算机科学与工程学院。

Robert C. Miller（rcm@mit.edu）来自马萨诸塞州剑桥MIT CSAIL。

David R. Karger（karger@mit.edu）来自马萨诸塞州剑桥MIT CSAIL。

David Crowell（dcrowell.mit@gmail.com）

Katrina Panovich（kpan@google.com）来自加利福尼亚州山景城谷歌公司。

译文责任编辑：唐杰

在本《通讯》独家视频中 http://cacm.acm.org/videos/soylent-a-word-processor-with-a-crowd-inside，您可以观看作者对本研究的讨论。