DOI:10.1145/2736282

归纳编程可以让用户从乏味的重复性任务中解放出来。

撰稿人：SUMIT GULWANI、JOSÉ HERNÁNDEZ-ORALLO、EMANUEL KITZELMANN、STEPHEN H. MUGGLETON、

归纳编程走入现实世界

世界上有许许多多人利用电脑完成日常的任务，但大部分因为缺少编程技能而不能从计算能力中获益。最为重要的是，由于用户不知道如何使用许多应用程序中提供的宏语言，他们常常不得不手动执行重复性操作。近期，大众市场首款产品以 Microsoft Excel 2013 中的“快速填充”(Flash Fill) 功能的形式诞生。借助“快速填充”，用户可以从自己提供的一个或多个示例为电子表格生成字符处理程序。由于融入了归纳编程方法，“快速填充”可以从少许的示例中习得许多较为复杂的程序。

归纳编程 (IP) 是一项横跨计算机科学、人工智能和认知科学的跨学科研究，主要研究如何从示例和背景知识中自动合成计算机程序。从归纳程序合成的研究发展而来的 IP 现在被称为*归纳功能编程*（*Inductive Functional Programming，IFP*），而从逻辑归纳推理技术发展而来的 IP 则被称为*归纳逻辑编程*（*Inductive Logic Programming，ILP*）。IFP 解决的是递归性功能程序的合成，它们利用生成与测试方法从输入/输出示例[19,41] 中检测（跟踪发现）的规律总结而来，其基于开创性 [27,34,35] 或系统性[16,28]搜索或数据驱动型分析方法。[6,11,17,23,38] 它的开发是利用演绎和形式化方法从完整规范中合成程序的工作的补充。[8]

ILP 来源于对逻辑框架[30,39] 中归纳的研究，受到了来自人工智能、机器学习和关系数据库的影响。它是拥有自己的理论、实施和应用的成熟领域，最近以一系列国际会议的形式庆祝了其 20 周年诞辰[33]。

在过去十年中，IP 已经吸引了国际上众多研讨会的目光。近期的调查[7,13,18] 反应了该研究领域广泛多样的实现和应用。

重要见解

- **帮助最终用户利用计算机自动执行复杂的重复性任务是一项巨大挑战，需要实现技术新突破。**
- **在软件应用程序中集成归纳编程技术，从观察用户与系统之间的交互来习得领域特定程序，从而为用户提供支持。**
- **归纳编程正在转移到现实应用程序，如电子表格工具、脚本编写和智能程序辅导等。**
- **与标准的机器学习相比，由于用户和系统共享相同的背景知识，归纳编程中能够从少量示例进行学习。**

图片由 PAVEL IGNATOV 提供

在最终用户编程领域中，人们提出了通过观察用户的输入行为来习得小例程的演示编程方法。[5,22,24] Excel 的“快速填充”功能提供了让人印象深刻的演示，可以成功应用 IP 中开发而来的程序合成方法为最终用户编程提供更多灵活性和动力。[9,11] 进一步的应用在其他桌面应用程序（如 ConvertFrom-String cmdlet[2] 中的 PowerShell 脚本）以及家用机器人和智能手机等专用设备中实现。

在本文中，我们将选择几个当代应用进行介绍。我们将 IP 的特征与典型机器学习方法进行对比，展示 IP 模型与人类归纳学习的认知模型之间有怎样的关系。最后，我们将探讨拓展 IP 的范畴和能力的最新技术（如对领域特定语言和元级学习的使用），以及面临的新挑战。

现实应用

最初，IP 应用于合成面向普通用途任务的功能或逻辑程序，例如操控数据结构等（如对列表排序或反转）。这些研究表明能够从少许输入/输出示例中合成小程序。近期的 IT 革新已经为此类技术创造了现实机会。当今大量的计算机用户中大部分都不是程序员，只能有限地利用为他们提供的软件。IP 可以武装这一类用户，让他们更加有效地利用计算机来自动化日常重复性任务。我们在此探讨一些此类机遇，尤其是最终用户编程和教育的领域中。

最终用户编程。计算设备最终用户常常需要创建小脚本（或许是一次性脚本）来自动执行重复性任务。这些用户可以利用示例轻松指定自己的意图，而这正是 IP 的用武之地。例如，可以考虑数据操控的领域。文本/日志文件、电子表格和网页等各种类型的文档通过将呈现和格式与底层数据模型相结合，为它们的创造者在存储和组织层次数据方面提供极大的灵活性。不过，这使提取用于常见任务的底层数据变得特别困难，如数据处理、查询、改变呈现视图或将数据转换为另一种存储格式。

数据操控的现有程序性解决方案（如 Excel 宏语言、Perl/Python 中的正则表达式库，以及 JavaScript 的 JQuery 库）存在三大局限。首先，这些解决方案针对特定领域，而且不同的文档类型需要不同的技术专长。其次，它们要求了解整个底层文档结构，包括最终用户不感兴趣的数据字段（其中一些在文档显示层中或许也不可见）。第三，也是最重要的，它们需要编程知识。因此，用户必须通过手动方式执行重复性任务，既耗费时间又容易出错。

归纳合成有助于解决各种数据操控任务。例如：(a) 从文本文件、网页和电子表格等半结构化文档提取数据[23]（如图 1 所示）。(b) 转换原子数据类型，如字符串[9]（如图 2 所示）或数字。转换复合数据类型，如表格[11] 和 XML。[35] (c) 格式化数据。[36] 将这些技术组合到提取、转换和格式化的管线之中，可以让最终用户能够执行复杂的数据操控任务。

计算机辅助教育。人类学习和交流通常围绕示例来构建 — 无论是学生通过示例理解或掌握某种概念，还是教师通过示例行为了解学生的错误想法或提供反馈。在归纳合成范畴中发展的基于示例型推理技术，可以为自动化教育领域中多种重复性结构化任务提供帮助，如问题生成、解决方案生成以及反馈生成等。[10] 这些任务可以针对广泛的 STEM 主题领域进行自动化，如逻辑、自动机理论、编程、算术、

图 1.FlashExtract[23]

利用示例从文本文件和网页等各种文档提取数据的框架。用户突出显示 (a) 中文本文件的各个字段的一两个示例时，FlashExtract 提取更多此类实例并将它们以结构化格式整理在 (b) 中的表格内。其实现方式是采用 (c) 中的领域特定语言 (DSL) 合成程序（与 (a) 中的示例相一致），然后再对 (a) 中的文本文件执行该程序。

Ana Trujillo
357 21st Place SE
Redmond, WA
(757) 555-1634

Antonio Moreno
515 93th Lane
Renton, WA
(411) 555- 2786

Thomas Hardy
742 17th Street NE
Seattle, WA
(412) 555-5719

Christina Berglund
475 22th Lane
Redmond, WA
(443) 555-6774

Hanna Moos
785 45th Street NE
Puyallup, WA
(376) 555-2462

Frederique Citeaux
308 66th Place
Redmond, WA
(689) 555-2770

(a)

标签 1	标签 2	标签 3
Ana Trujillo	Redmond	(757) 555-1634
Antonio Moreno	Renton	(411) 555-2786
Thomas Hardy	Seattle	(412) 555-5719
Christina Berglund	Redmond	(443) 555-6774
Hanna Moos	Puyallup	(376) 555-2462
Frederique Citeaux	Redmond	(689) 555-2770

(b)

PairSeq SS ::= LinesMap(λx:Pair(Pos(x, p_1), Pos(x, p_2)), LS)
| StartSeqMap(λx:Pair(x, Pos($R_0[x], p$)), PS)
LineSeq LS ::= FilterInt(init, iter, BLS)
BoolLineSeq BLS ::= FilterBool(b, split(R_0, ’/n’))
PositionSeq PS ::= LinesMap(λx:Pos(x, p), LS)
| FilterInt(init, iter, PosSeq (R_0, rr))
Pred b ::= λx: {Starts,Ends}With(r,x) | λx:Contains(r, k, x)

(c)

代数和几何等。例如，图 3 中显示了一种归纳合成技术的输出，它用于生成类似于示例题目的代数证明题。

未来机遇。我们已介绍了 IP 的重要现实应用。我们相信，不久的将来 IP 可以并且即将应用到许多其他领域。任何已存在一组高级抽象的领域都是 IP 的有力候选者。例如，若此即彼（If This Then That，IFTTT）服务 (http://ifttt.com/) 允许最终用户使用触发器和操作来表述基于规则的小程序，可谓 IP 应用的绝佳候选者。IFTTT 通过特定的渠道（如 Facebook）将触发器（如：*我在照片中圈出*）与操作（如：*发送电子邮件*）连接。在这种领域中，IP 可以从用户执行任务的示例中习得程序。比如，它可以习得这样的程序：每当智能手机用户离家上班时发送短信。展望未来，从用户提供的示例构建机器人策略[30] 将是 IP 前途光明的新方向。

随着构建基于 IP 的解决方案的框架发展成熟，如简化合成器构建流程的元数据合成框架（稍后会探讨），开发人员将越来越轻松地创建由 IP 支持的新应用程序。

IP 与机器学习对比

IP 在于让机器自动习得程序，所以可被视为另一种机器学习范例。那么，归纳编程有什么特别之处？表 1 概述了一序列差别，我们将对其中一些进行探讨。

我们也会借助一个运行示例来指出 IP 的一些特征。图 4 展示了一款说明性应用程序，其目标是识别有关用户个人联系人的重复模式和规则。IP 系统习得一条简单规则（用户的上司必须添加到她的圈子中）和一条较为复杂的规则（与家庭成员结婚的任何人也应当添加到她的圈子中）。从中我们可以看到一些 IP 系统的特征，如少量的示

图 2.快速填充[9]

此为 Excel 2013 的一项功能，可以通过示例自动执行重复性字符转换。用户执行所需转换的一个实例（行 2，列 B）并继续转换另一实例（行 3，列 B）后，“快速填充”习得一个程序 `Concatenate(ToLower(Substring(v,WordToken,1))," ", ToLower(SubString(v,WordToken,2)))`，此程序从输入字符串 v（列 A）提取前两个单词，将它们转换为小写格式，然后连接起来并用一个空格字符分隔，从而自动执行这一重复性任务。

	A	B
1	Email	Column 2
2	Nancy.FreeHafer@fourthcoffee.com	nancy freehafer
3	Andrew.Cencici@northwindtraders.com	andrew cencici
4	Jan.Kotas@litwareinc.com	jan kotas
5	Mariya.Sergienko@gradicdesigninstitute.com	mariya sergienko
6	Steven.Thorpe@northwindtraders.com	steven thorpe
7	Michael.Neipper@northwindtraders.com	michael neipper
8	Robert.Zare@northwindtraders.com	robert zare
9	Laura.Giussani@adventure-works.com	laura giussani
10	Anne.HL@northwindtraders.com	anne hl
11	Alexander.David@contoso.com	alexander david
12	Kim.Shane@northwindtraders.com	kim shane
13	Manish.Chopra@northwindtraders.com	manish chopra
14	Gerwald.Oberleitner@northwindtraders.com	gerwald oberleitner
15	Amr.Zaki@northwindtraders.com	amr zaki
16	Yvonne.McKay@northwindtraders.com	yvonne mckay
17	Amanda.Pinto@northwindtraders.com	amanda pinto

图 3.通过分析函数进行代数证明问题的生成。

给定的问题普遍化为一个模板，通过对自由变量测试随机值来找到有效的实例化。

示例问题 ↓	$\frac{\sin A}{1+\cos A} + \frac{1+\cos A}{\sin A} = 2\csc A$
泛化问题模板 ↓	$\frac{T_1 A}{1 \pm T_2 A} + \frac{1 \pm T_3 A}{T_4 A} = 2\ T_5 A$ 其中 $T_i \in \{\cos, \sin, \tan, \cot, \sec, \csc\}$
新相似问题	$\frac{\cos A}{1-\sin A} + \frac{1-\sin A}{\cos A} = 2\tan A$
	$\frac{\cos A}{1+\sin A} + \frac{1+\sin A}{\cos A} = 2\sec A$
	$\frac{\cot A}{1+\csc A} + \frac{1+\csc A}{\cot A} = 2\sec A$
	$\frac{\tan A}{1+\sec A} + \frac{1+\sec A}{\tan A} = 2\csc A$
	$\frac{\sin A}{1-\cos A} + \frac{1-\cos A}{\sin A} = 2\cot A$

表 1：归纳编程与其他机器学习范例的简单比较。

	归纳编程	其他机器学习范例
示例数	小	大，如大数据
数据种类	关系，基于构造器的数据类型	普通表格，序列数据
数据源	人类专家、软件应用程序、HCI，等等	关系数据库、互联网、传感器 (IoT)，等等
假说语言	说明性：通用编程语言或领域特定语言	线性、非线性、基于距离、基于内核、基于规则、概率性，等等
搜索策略	提炼、抽象运算符、穷举。	梯度下降、数据分区、覆盖、基于实例，等等。
表示法学习	高阶和预测/函数发明	深层学习和特征学习。
模式可理解性	寻常。	不寻常。
模式可表达性	通常为递归，甚至图灵完备。	特征值，非图灵完备。
学习偏倚	使用背景知识和约束	使用先验分布、参数和特征。
评估	多样标准，包括简易性和可理解性。	以最小化错误（或损失）为方向。
验证	代码检查、分治法调试、背景知识一致性	统计论证（仅少数技术可以本地检验）。

例、数据的类型和来源、背景知识的角色、用户的交互和反馈、通用说明性语言的使用、递归的使用，以及学习模式的可理解性和表达性。尽管机器学习和 IP 是比较互补的关系，它们也可以良好地搭配。例如，如果 IP 生成了多个与提供的示例相符的程序，便可使用机器学习对这些程序进行排名。[11]

小数据。随着数据的收集与存储成本变得越来越低，人们倾向于认为现在唯一让人感兴趣的数据集是*大数据*。然而，来自单一用户与任何设备的交互的数据集通常非常小，例如图 4 顶部所示的针对个人日程收集的数据量。

众所周知，相较于从大量数据学习，从少量示例中学习比较困难也不可靠。示例越少，我们就越容易过度拟合，特别是使用表达性语言时。IP 在示例少、假说空间大时特别有用（图灵完备）。

说明性表示法。大多数（统计上）机器学习技术基于概率、距离、重量、内核和阵列等。这些方法中除了基于（命题的）决策树和规则以外，其余都不是*说明性*的，即表述为易于理解的规则。因此，IP 的另一个显著特征是它利用丰富的符号表示法，因为假说通常是*说明性*的程序。

说明性方法允许使用单一语言表示背景知识、示例和假说，如图 4 中所示。除了（最终）用户可使用单一语言的便利之外，知识的检验、修改以及与其他知识来源整合也变得更加简单。因此，递增、累积或持久的学习变得更加容易。[13] 例如，NELL（Never-Ending Language Learner，无止境语言学习器）[3] 利用 ILP 算法学习概率性的霍恩子句。

现在，IP 中的许多语言都是*混合型*的，如功能逻辑编程语言，以及具有类型和高阶构造、约束和可能性的逻辑编程语言等。逻辑 (ILP) 与功能 (IFP) 之间的论战最近也因为领域特定语言（Domain-Specific Language，DSL）的突破而得以平息，后者通常更加适合目前的应用，我们稍后会予以阐述。

提炼与抽象。与 IP 相关的另一个特定问题是正确组合多种推理机制（如推演、溯因推理和归纳）来组织假说空间的方式。许多早期的 IP 运算符是推演运算符的反演，形成自下而上和自上而下的方法，它们分别使用普遍化和专门化运算符。[33] 更广泛地来说，可以根据语言的运算语义来定义*提炼*和*抽象*运算符，包括使用高阶函数、断言和函数创造。

这在仅仅外延的事实和更有内涵的知识之间建立了许多层面，形成一种层次结构。实际上，对事实、背景知识和假说使用同一种表示语言可以促进这种层次结构，如图 4 中所示。

深层知识。由于抽象机制和对背景知识的利用，IP 将学习视为一种知识获取过程。例如在图 4 中，归纳编程可以访问关于联系人组和联系人之间关系（例如家庭关系或单位组织结构）的一些信息。此类知识称为*背景知识*，充当有力的显式偏倚，从而缩小搜索空间并找到正确的普遍化层级。

如果知识引用了较低层级的定义（包括递归引用其自身），则可被视为*深层*知识。对这种结构化深层知识的表示通过编程语言来实现，这些编程语言具有变量、丰富的运算语义，以及更为重要的*递归性*。递归是归纳编程中的一个关键。[19,30,38,41]请注意，图 4 示例中的背景知识和新假说都具有递归性。

这与其他机器学习方法中背景知识仅有先验分布、概率或特征的形式恰恰相反。与深层学习的其他非象征方法[1] 也存在显著区别；深层学习是机器学习中的一种新方法，也以层次结构方式构建比较复

杂的模型和特征，但数据、知识和偏倚的表示都不同。

目的和评估。在其他机器学习方法中，通过计入偏差率的不同指标来衡量假说。IP 的目的不仅仅是最大化一些特定的错误指标，而是要根据 IP 应用的用途找到具有可操作性的有意义程序。这通常暗示着它们必须与数据的绝大部分（若不是全部）相符，而且也要与背景知识和其他的约束相符。此外，由于假说是说明性的（可能也是递归性的），其评估比较多样化，包括简易性、可理解性、一贯性和时间/空间复杂性等标准。

最新技术

IP 本质上是一种搜索问题，可以从各个领域中开发的技术获益。我们在此这展示几种在最新 IP 研究中使用的技术。

DSL 合成器。DSL 基于下列方法学引入到 IP 情景中：

*1. 问题定义：*通过研究帮助论坛和执行用户研究，确定任务垂直领域并收集通用情景。

*2. 领域特定语言：*设计一种 DSL，其表现力足以捕获相关领域中的现实任务，并且其局限性也足以从示例实现有效的学习。图 1(c) 介绍了这样的一种 DSL，它可以从文本文件提取数据。（此 DSL 的完整版本及其语义学在 Le 和 Gulwani 的论文中介绍。[23]）这一 DSL 可以通过结合使用 filter 和 map 运算来提取一序列子字符串。

*3. 合成算法：*这些算法中大部分的原理为，应用分而治之的推演技术系统地将问题从原始表达式缩小到子表达式的组合（通过将表达式的示例转换为子表达式的示例）。这些算法通常以计算一组 DSL 程序而告终。

*4. 排名：*可能使用机器学习技术，对合成器返回的各种程序进行排名。

这种方法学已应用到各式各样的领域，包括转换句法字符串、[9,29] 语义字符串、数字和表格。[11]

综合集成框架。领域特定合成器（与通用合成器相对）在效率（例如快速合成程序的能力）和排名（例如从较少的示例合成所需的程序的能力）方面具有诸多优势。但是，领域特定合成器的设计和开发是一个不寻常的过程，需要重要的领域见解和实施工作。此外，对 DSL 的任何更改也需要对合成器进行大量的更改。

综合集成框架可以简化合成器的开发，以利用同一组合器核心集合构建的 DSL 相关系列。构建此类

图 4.IP 系统交互示例以及背景知识的关键作用。

用户就其日程和联系人组与多个设备交互。系统收集一序列事实、规则和运算符，形成背景知识。从背景知识和少许示例中，系统能够推导出新的规则，经验证后用于扩充背景知识，并且为用户提供建议或给出建议选项。

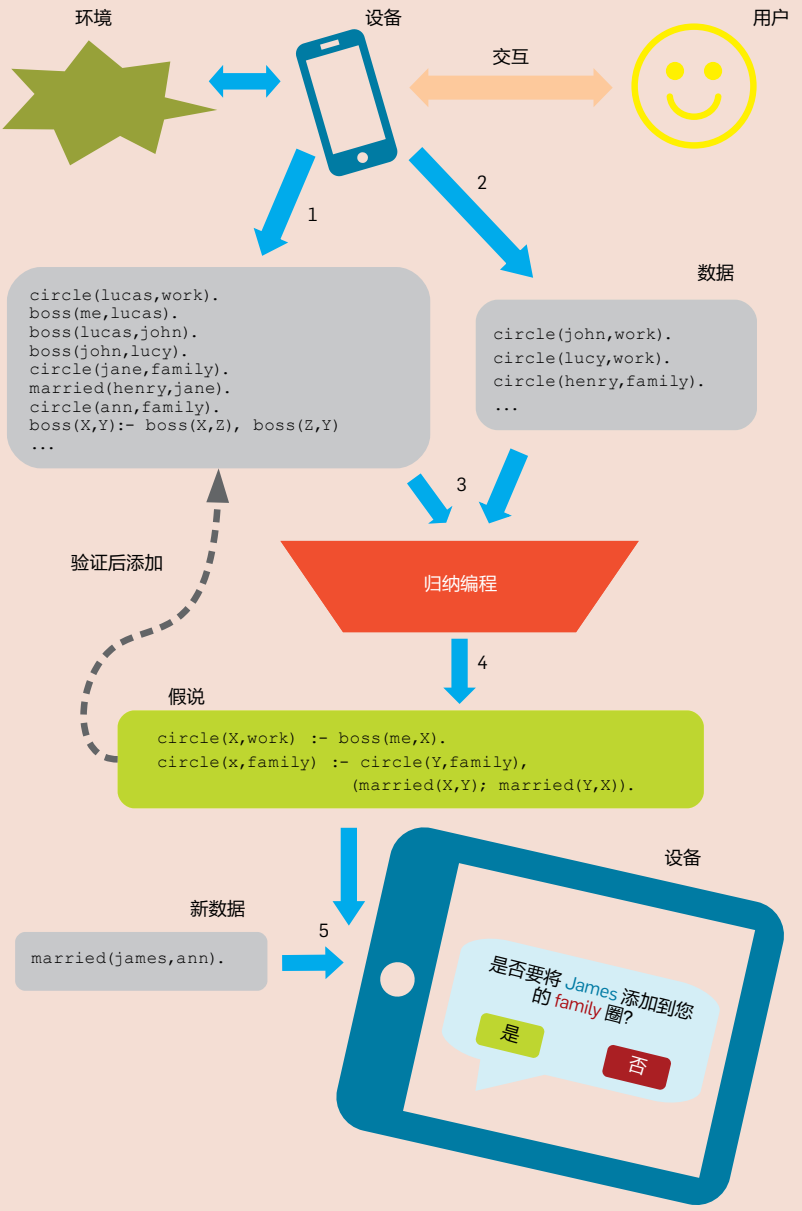


图 5：给定将长度零到三的列表颠倒的示例，Igor2 利用高阶函数 fold 合成程序。

辅助函数 *f* 在列表的末尾追加一个元素，并对 fold 进行参数化，此函数不是背景知识而是生造而来。

```
reverse x = fold f [] x

f x0 [] = [x0]

f x0 (x1 : xs) = x1 : f x0 xs
```

表 2：对图 4 中 boss 断言使用 chain 假说来利用元规则和背景知识证明新示例 boss(lucas, lucy)。

名称	元规则
Chain	$P(x,y) \leftarrow Q(x,z), R(z,y)$
示例	**背景知识**
boss(lucas, lucy)	boss(lucas, john). boss(john, lucy).
元替代	**Chain 假说**
P=Q=R=boss	boss(X,Y) :- boss(X,Z), boss(Z,Y)

框架涉及下列步骤：

1. 确定一组允许通用用户交互模型的垂直任务领域。

2. 设计 DSL 的代数。DSL 是语法规则的一个有序集合（模型排名）。

3. 为各个代数算子设计一个搜索算法，使之具有组合性和归纳性。

综合集成框架可以让合成器作者轻松开发领域特定合成器，这与 lex 和 yacc 等说明性解析框架允许编译器作者轻松编写解析器相似。FlashExtract 框架[23]和测试驱动型合成框架[35] 有助于轻松开发用于从各种文档类型（如文本文件、网页、XML 文档、表格和电子表格等）提取和转换数据的合成器。图 1(c) 说明了由 Filter 和 Map 运算符组成的 DSL，它由 FlashExtract 框架提供支持。

所以，FlashExtract 框架能够为这一 DSL 自动构造有效的合成算法。

高阶函数在搜索假说时可能提供偏差。与 DSL 相对，高阶函数不会针对预定义的领域定制 IP，而是为处理递归（关联）数据提供通用模式，作为 IP 系统的背景知识。例如，`fold` 高阶函数（也称为 *reduce*）迭代一个元素列表，并通过应用另一个函数（它也作为参数提供给 `fold`）结合这些元素。例如，`fold (+) [1, 2, 3, 4]` 可以通过加号结合列表中的数字，并返回 10。然后，IP 不必习得递归函数，而仅需要挑选合适的高阶函数并适当地进行实例化。在 IP 中利用高阶函数的首批系统中有一个便是 MAGICHASKELLER[16]，它从一小组正数输入中生成 Haskell 函数。生成的程序是 fold 等一组预定义高阶函数的实例化。分析 IP 系统扩展 Igor2 也在 Haskell 实施，它采用了类似的方法。与通过枚举查找程序的 Katayama[16] 相对，Igor2 通过分析给定的数据来决定适合的高阶函数。用于实例化高阶函数的辅角函数从背景知识中选取，如果不存在的话，则作为辅助函数而生造。使用这一技术不仅能够加快合成速度，而且也能扩大可合成程序的范围。[14] 图 5 中提供了利用高阶函数进行归纳的示例。最后，Henderson[12] 提出了使用高阶函数限制和引导程序搜索以进行累积学习的建议，其中对归纳自示例的函数进行抽象，然后用于归纳更为复杂的程序。与 DSL 不同，高阶并不将 IP 限制为某一预定义的领域，相反它的作用是引导搜索。

元解释学习（Meta-interpretive Learning，MIL）是新近出现的一种 ILP 技术，[31,32] 它旨在支持对递归定义的学习。MIL 有一个强大而新颖的方面，在学习断言定义时它可自动引入子定义，能够分解为由可重用部分组成的层次结构。MIL 基于 Prolog 元解释器的修改版本。通常，此类元解释器通过重复获取一阶 Prolog 子句（其标头通过给定目标而统一）来衍生出证明。相反，元解释学习程序额外获取高阶元规则（其标头通过目标而统一），再保存生成的元替代项以形成程序。为说明这一理念，我们可以思考表 2 中与 *P*、*Q* 和 *R* 相关的元规则。在本例中，通过利用元规则提供示例和背景知识从元替代至 *X*、*Y* 和 *Z* 元变量中习得对图 4 中 boss 断言的 chain 假说。

给定了高阶替代时，可以重新构建实例化的程序子句，并在后面的证明中使用，从而形成一种支持自动构建层级化定义程序的IP形式。

在 Lin 等人[25] 的文章中，作者将 MIL 应用到设计 Gulwani[9] 曾经研究过的字符串转换任务。图 6 显示了应用 MIL 来学习一组此类任务的结果，它采用了*依赖*学习和*独立*学习这两种方法；在前者中，新的定义可以在更低的层次调用已经习得的定义。依赖学习生成更为紧凑的程序，因为它重新利用了现有的子定义。这反过来也能减少搜索次数，因为查找较小的任务定义需要的搜索工作也较少。

约束求解。此处的总体思路是将合成程序缩小为相当的满足性程序，表达为标准的逻辑公式。然后，这一公式通过一般的现成工具求解，利用可满足性 (SAT) 和可满足性模理论 (SMT) 求解程序的最新技术进步。此方法已应用于从完整的形式规范从合成，但其应用性被限制为合成受限形式的程序。另一方面，如果规范是示例的形式，那么缩减合成问题来求解 SAT/SMT 约束就可以对更大范围的程序执行。这些示例或许可以在以计数器示例引导的归纳合成循环[40]（这涉及在当前版本的合成程序不满足给定规范时利用验证技术来查找新的测试输入）中生成，或者利用基于区分性输入的方法学[15]（设计查找可区分两种语义不同、但都与给定的示例组相符的合成程序的新测试输入）。

挑战

人们在 IP 中持续不断地投入研究，以解决大小、效用和鲁棒性等方面挑战性愈来愈强的难题。

复合性。IP 若要能够妥当地执行更为复杂的任务，需要在多个方面实现突破。首先，正确求解的搜索空间的底层复杂性限制了 IP 的总体可用性，尤其是在需要即时反馈的交互环境中。此类方法的性能上无疑会有进步，比如版本空间代数等提供搜索空间紧凑表示法的各种方法。最终，将存在任何算法改进都无法解决的复杂性限制。在这种情形中，需要能够让用户将较复杂的任务分解为足够小的子任务、然后以递增方式为各项子任务编写由 IP 提供的求解程序的新方法。

新型穷举搜索。此处的总体思路是，系统地探索工件的整个状态空间，并且对照给定的示例检查各个候选者的正确性。当规范由示例组成（与形式关系规范相反）时，这种方法相对而言效果不错，因为针对示例检查候选求解程序的正确性要比针对形式关系规范验证正确性来得快得多。然而，这说起来容易做起来难。因为潜在工件拥有巨大的底层状态空间，通常需要创新且不凡的优化，例如以目标为导向的搜索、分支定界、以复杂性为引导的创新方法、基于示例的文本特征的线索，[28] 以及脱机索引。[16]

领域变化。高效地将 IP 应用到新的领域也需要新的方法，包括创建前面提到过的元合成器。由于在现实应用中应用 IP 技术相对较新，探索应用空间的经验尚且不足，难以识别可能从不同领域中产生的通用模式。或许在短期内，会以专设的方法开发领域特定 IP 系统；随着时间推移，此类系统的经验得到积累后，新的方法将能够系统化和形式化这些专设的做法，系统因而也能变得更加通用，可以在不同的领域之间重复利用。

验证。IP 产生的工件一定要给予最终用户信心，让他们相信自己创建的内容是正确的、有意义的。例如，由元解释学习（如前文所述）等方法生成的新生子任务过多地采用自动命名的层次结构时，如果新名称不能与所定义的子任务的语义之间存在清晰的对应关系，那就可能会造成混淆。要解决这样的挑战，我们必须找到新的方法，以浅显直观的术语向用户说明生成的程序的行为，并为他们寻找方法来引导求解（如果不正确的话）。这一问题上有着非常的大的创造空间。例如，使用将用户与 IP 结果衔接的抽象，当工具信心度欠佳并且用户应当考虑检查结果时突出显示相关输入的功能，明显展现合成的程序以领域特定的直观方式（如使用图像）来应用，并且以自然语言重新组织合成程序的语句，以及让用户进行风格化的编辑。

噪声容限。真实的数据通常往往可能存在偏差，一些值可能缺失且/或不正确，另一些值可能以不同的格式出现（例如在表示日期和数字时）。有时候，如果用户提供时意外出错，背景知识甚至也可能不正确。

解决对抗此类噪声的鲁棒性问题或许要以领域特定的方式来进行。例如，如果一张表格中的数据大部分都正确，但具有少量的异常，检测与报告离群值（或者甚至

图 6：对利用 MIL 由依赖学习和独立学习生成的程序进行对比。

标有 *n* 的节点对应于求解任务 *n* 的程序，节点按照其大小进行垂直排列。对于依赖学习（左侧），箭头对应于归纳程序的调用关系。依赖学习产生对现有子定义的再利用，这反过来也可减少搜索次数。

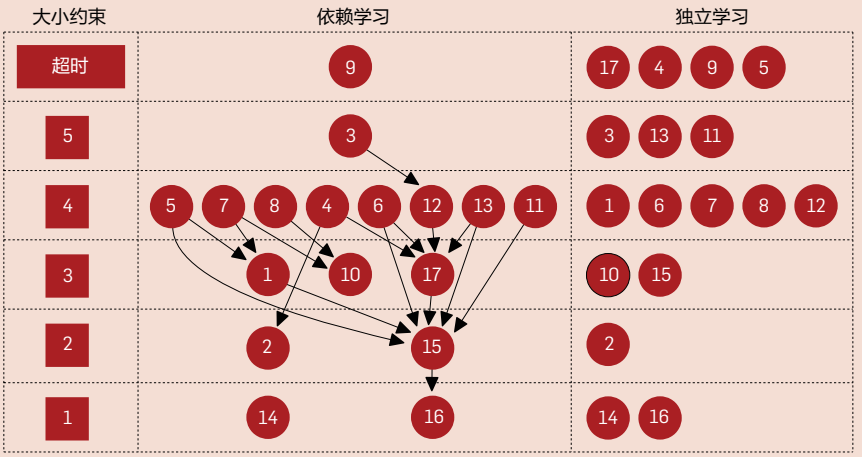
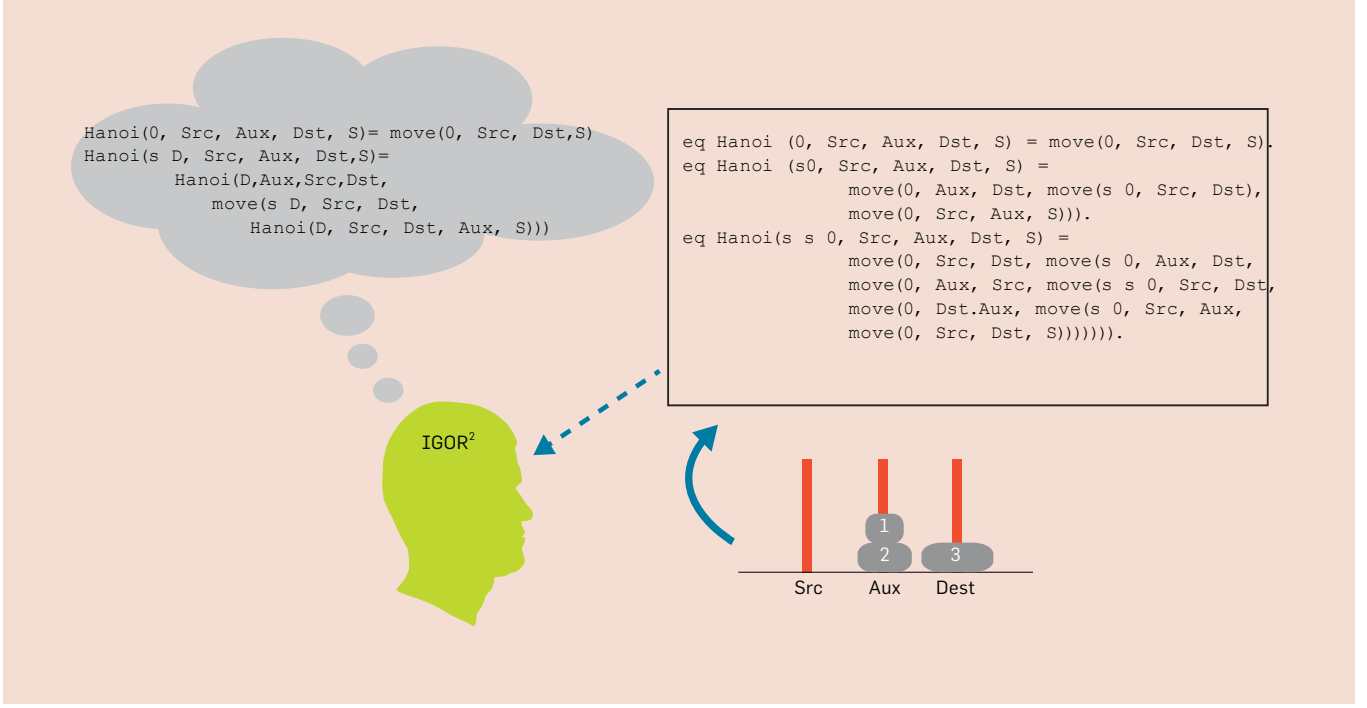


图 7.Igor2 的 *Tower of Hanoi* 问题输入，以及归纳的递归规则集。



缺失的值）的现有技术能为 IP 过程助力。幸运的是，现有的 ML 范畴中已经有大量的研究，可以应用到这一问题（例如，可参见 Chandola 等人[4] 的文章）。

使 IP 更具认知性。认知科学和心理学表明，人类从少量的（通常正面的）示例中学习，相对而言难以容忍异常。[26] 一贯性、简易性和解释能力是人类归纳推理的引导规则。认知科学中也研究了背景知识的角色，以及为获取更加抽象的概念而需要开发的必要构造。[42]人类循序渐进获取深层知识尤其体现在语言学习中，但也体现于从问题解决经验中学习上。最近，IP 已经用于认知建模的范畴中，显示出可以仅从少量的正例中习得普遍化的规则。[37] 例如，图 7 显示了 IP 系统 Igor2 归纳的 *Tower of Hanoi* 问题学习结果。此结果相当于（一些）人从相同示例推导的三个圆盘至 *n* 个圆盘问题的普遍化。[20]然而，到目前为止，依然没有实证研究可以在人类对复杂例程的高级学习和培训投入与 IP 系统的归纳程序之间进行详细的比较，从而能了解它们是否可形成相似的解决方案以及它们何时还有分歧，并且了解 IP 解决方案是否依然能为人类理解。

假如 IP 系统中学习的说明性实质与人类的知识级学习足够相似，IP 系统便可通过感知用户界面[43] 来增强现实以实现机器如人类一样交互的终极目标，并且评估它们是否能够以这种方式变得更加直观、可信、熟悉并且可预测（包括预测到系统即将失败）。为了能通过 IP 实现这一目标，我们需要解决*交互模型*。例如，来自用户的监管可被限制为对系统操作的一些奖励（“确定”按钮）或惩罚（“取消”按钮），如图 4 中所示。另外，用户也可提供少许示例，IP 系统猜测其他示例，用户再进行更正。[5,11] 在这一互动（或查询）学习过程中，用户可以利用用户提供的（或者由系统本身提供，[15] 这更加有效）*区分输入*来显示它们之间的区别，以便从一组候选假说中进行挑选。

结论

自上世纪七十年代起，IFP 和 ILP 中的基本研究推动了基础算法的发展以处理从输入/输出示例归纳程序的问题。然而，这些方法保留在人工智能研究的范畴之内，未能触发成功转变为适用范围更加广泛的技术。2009 年，Tessa Lau 通过演示系统展示了重要的编程探讨，表示此类系统的应用尚未得到广泛分布，并且提出这主要是因为此类系统缺乏可用性。[21] 在本文中，我们也展现了 IP 相关的近期研究，我们确定了一些新的方法和技术，它们有潜力攻克前代系统的一些局限：当用户和系统共享能够以说明性方式表示的背景知识时，可以实现从极少量正例中学习，与名称推断相结合时也可能更加容易理解。利用在 ILP 和/或 IFP 中开发的算法技术，以及运用更为强大的 IP 算法中生成的高阶函数和元解释学习，采用基于领域特定语言的技术已经能够实现可运用于大众市场产品（如“快速填充”）的技术。近期的成果有望从 IP 源自的各个不同领域（人工智能、机器学习、功能编程、ILP、软件工程以及认知科学）吸引更多的研究人员来攻克让 IP 从实验室走向现实世界的难题。

参考资料
1. Bengio, Y., Courville, A. and Vincent, P. Representation learning:A review and new perspectives.*Pattern Analy. Machine Intell.35*, 8 (2013), 1798–1828.
2. Bielawski, B. Using the convertfrom-string cmdlet to parse structured text.*PowerShell Magazine*, (Sept. 9, 2004); http: // www. powershellmagazine. com/ 2014/ 09/09/ using-the-convertfrom-string-cmdlet-to-parse-structured-text/
3. Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka-Jr, E.R. and T.M. Mitchell, T.M. Toward an architecture for never-ending language learning.In *AAAI*, 2010.
4. Chandola, V., Banerjee, A. and V. Kumar, V. Anomaly detection:A survey.*ACM Computing Surveys 41*, 3 (2009), 15.
5. Cypher, A. (Ed).*Watch What I Do:Programming by Demonstration*.MIT Press, Cambridge, MA, 1993.
6. Ferri-Ramírez, C., Hernández-Orallo, J. and Ramírez-Quintana, M.J. Incremental learning of functional logic programs.In *Proceedings of FLOPS*, 2001, 233–247.
7. Flener , P. and Schmid, U. An introduction to inductive programming.*AI Review 29*, 1 (2009), 45–62.
8. Gulwani, S. Dimensions in program synthesis.In *Proceedings of PPDP*, 2010.
9. Gulwani, S. Automating string processing in spreadsheets using input-output examples.In *Proceedings of POPL*, 2011; http://research.microsoft.com/users/sumitg/flashfill.html.
10. Gulwani, S. Example-based learning in computer-aided STEM education.*Commun.ACM 57*, 8 (Aug 2014), 70–80.
11. Gulwani, S., Harris, W. and Singh, R. Spreadsheet data manipulation using examples.*Commun.ACM 55*, 8 (Aug. 2012), 97–105.
12. Henderson, R.J. and Muggleton, S.H. Automatic invention of functional abstractions.*Latest Advances in Inductive Logic Programming*, 2012.
13. Hernández-Orallo, J. Deep knowledge:Inductive programming as an answer, Dagstuhl TR 13502, 2013.
14. Hofmann, M. and Kitzelmann, E. I/O guided detection of list catamorphisms—towards problem specific use of program templates in IP.In *ACM SIGPLAN PEPM*, 2010.
15. Jha, J., Gulwani, S., Seshia, S. and Tiwari, A. Oracle-guided component-based program synthesis.In *Proceedings of the ICSE*, 2010.
16. Katayama, S. Efficient exhaustive generation of functional programs using Monte-Carlo search with iterative deepening.In *Proceedings of PRICAI*, 2008.
17. Kitzelmann, E. Analytical inductive functional programming.*LOPSTR 2008, LNCS 5438*.Springer, 2009, 87–102.
18. Kitzelmann, E. Inductive programming:A survey of program synthesis techniques.In *AAIP*, Springer, 2010, 50–73.
19. Kitzelmann, E. and Schmid, U. Inductive synthesis of functional programs:An explanation based generalization approach.*J. Machine Learning Research 7*, (Feb. 2006), 429–454.
20. Kotovsky, K., Hayes, J.R. and Simon, H.A..Why are some problems hard?Evidence from Tower of Hanoi. *Cognitive Psychology 17*, 2 (1985), 248–294.
21. Lau, T.A. Why programming-by-demonstration systems fail:Lessons learned for usable AI.*AI Mag.30*, 4, (2009), 65–67.
22. Lau, T.A., Wolfman, S.A., Domingos, P. and Weld, D.S. Programming by demonstration using version space algebra.*Machine Learning 53*, 1-2 (2003), 111–156.
23. Le, V. and Gulwani, S. FlashExtract:A framework for data extraction by examples.In *Proceedings of PLDI*, 2014.
24. Lieberman, H. (Ed).*Your Wish is My Command:Programming by Example*.Morgan Kaufmann, 2001.
25. Lin, D., Dechter, E., Ellis, K., Tenenbaum, J.B. and Muggleton, S.H. Bias reformulation for one-shot function induction.In *Proceedings of ECAI*, 2014.
26. Marcus, G.F. The Algebraic Mind.*Integrating Connectionism and Cognitive Science*.Bradford, Cambridge, MA, 2001.
27. Martìnez-Plumed, C. Ferri, Hernández-Orallo, J. and M.J. Ramírez-Quintana.On the definition of a general learning system with user-defined operators. *arXiv preprint arXiv:1311.4235*, 2013.
28. Menon, A., Tamuz, O., Gulwani, S., Lampson, B. and Kalai, A. A machine learning framework for programming by example.In *Proceedings of the ICML*, 2013.
29. Miller, R.C. and Myers, B.A. Multiple selections in smart text editing.In *Proceedings of IUI*, 2002, 103–110.
30. Muggleton, S.H. Inductive Logic Programming.*New Generation Computing 8*, 4 (1991), 295–318.
31. Muggleton, S.H. and Lin, D. Meta-interpretive learning of higher-order dyadic datalog:Predicate invention revisited.*IJCAI 2013*, 1551–1557.
32. Muggleton, S.H., Lin, D., Pahlavi, N. and Tamaddoni-Nezhad, A. Meta-interpretive learning: application to grammatical inference.*Machine Learning 94* (2014), 25–49.
33. Muggleton, S.H., De Raedt, L., Poole, D., Bratko, I., Flach, P. and Inoue, P. ILP turns 20:Biography and future challenges.*Machine Learning 86*, 1 (2011), 3–23.
34. Olsson, R. Inductive functional programming using incremental program transformation.*Artificial Intelligence 74*, 1 (1995), 55–83.
35. Perelman, D., Gulwani, S., Grossman, D. and Provost, P. Test-driven synthesis.*PLDI*, 2014.
36. Raza, M., Gulwani, S. and Milic-Frayling, N. Programming by example using least general generalizations.*AAAI*, 2014.
37. Schmid, U. and Kitzelmann, E. Inductive rule learning on the knowledge level.*Cognitive Systems Research 12*, 3 (2011), 237–248.
38. Schmid, U. and Wysotzki, F. Induction of recursive program schemes.*ECML 1398 LNAI* (1998), 214–225.
39. Shapiro, E.Y. An algorithm that infers theories from facts.*IJCAI* (1981), 446–451.
40. Solar-Lezama, A. *Program Synthesis by Sketching*. Ph.D thesis, UC Berkeley, 2008.
41. Summers, P.D. A methodology for LISP program construction from examples.*J ACM 24*, 1 (1977), 162–175.
42. Tenenbaum, J.B., Griffiths, T.L. and Kemp, C. Theory-based Bayesian models of inductive learning and reasoning.*Trends in Cognitive Sciences 10*, 7 (2006), 309–318.
43. Young, S. Cognitive user interfaces.*IEEE Signal Processing 27*, 3 (2010), 128–140.

Sumit Gulwani (sumitg@microsoft.com) 是微软公司位于华盛顿州雷德蒙德的示例编程研究与工程团队首席研究员兼研究经理。

José Hernández-Orallo (jorallo@dsic.upv.es) 是西班牙巴伦西亚政治大学的讲师。他受到了欧盟 (FEDER) 和西班牙项目 PCIN-2013-037、TIN 2013-45732-C4-1-P及 GV PROMETEOII2015/013 的支持。

Emanuel Kitzelmann (ekitzelmann@gmail.com) 是位于德国拉廷根的 Adam-Josef-Cüppers 商学院的教师。

Stephen H. Muggleton (s.muggleton@imperial.ac.uk) 是英国伦敦帝国理工学院计算机系的教授。

Ute Schmid (ute.schmid@uni-bamberg.de) 是德国班贝克大学的教授。

Benjamin Zorn (Ben.Zorn@microsoft.com) 是微软研究院软件工程研究组 (RiSE) 的首席研究员和共同经理人，该研究院位于华盛顿市雷德蒙德。

译文责任编辑：田丰