基于高阶逻辑的复杂结构归纳学习研究*)

李琳娜　杨炳儒　周法国
(北京科技大学信息工程学院　北京 100083)

摘　要　归纳学习的目的在于发现样例与离散的类之间的映射关系，样例及归纳的映射都需用某个形式化语言描述。归纳学习器采用的形式化语言经历了属性-值语言、一阶逻辑、类型化的高阶逻辑三个阶段，后者能克服前二者在知识表达及学习过程中的很多缺点。本文首先阐述了基于高阶逻辑的复杂结构归纳学习产生的历史背景；其次介绍了基于高阶逻辑的编程语言——Escher的知识描述形式及目前已提出的三种学习方法；复杂结构的归纳学习在机器学习领域的应用及如何解决一些现实问题的讨论随后给出；最后分析了复杂结构归纳学习的研究所面临的挑战性问题。

关键词　归纳学习，高阶逻辑，归纳逻辑编程，遗传编程，复杂结构数据

Survey of Inductive Learning from Complex Structured Data Based on Higher-order Logic

LI Lin-na　YANG Bing-ru　ZHOU Fa-guo
(School of Information Engineering, Beijing University of Science and Technology, Beijing 100083, China)

Abstract　Inductive learning consists of finding mapping of examples into discrete classes. Examples and induced mapping are represented in some formal language. The formal language employed by inductive learners has undergone three stages, such as, attribute-value language, first-order logic, typed higher-order logic. The latter can overcome many shortcomings in knowledge representation and learning process which belong to the former two. This paper firstly provides a survey of background and context from which inductive learning from complex structured data based on higher-order logic arises. Secondly, how to represent knowledge by means of Escher, which is a typed higher-order logic programming language is demonstrated and three kinds of learning algorithm is introduced. The application of inductive learning from complex structured data based on higher-order logic in machine learning and how to solve some practical problems with which is discussed in following. Lastly, several challenging researching problems are identified.

Keywords　Inductive learning, Higher-order logic, Inductive logic programming, Genetic programming, Complex structured data

1　引言

归纳学习的目的在于发现样例与离散的类之间的映射关系，发现的映射关系不仅能准确地解释当前给定的样例，还要能准确地预测新的样例。样例及归纳的映射都需用某个形式化语言描述。历史上，属性-值学习(Attribute-Value Learning, AVL)使用的属性-值语言是机器学习领域使用最广泛的形式化语言。在AVL中，样例用属性值元组表示，每个属性表达样例的某项特征(如形状、颜色等)。AVL尽管很有用，但其自身也存在很多局限。近年来，研究者提出了用一阶逻辑作为机器学习中的形式化语言并由此产生了归纳逻辑编程(Inductive Logic Programming, ILP)[1]这样一个新的研究领域，很多ILP系统都采用PROLOG作为知识描述机制。

PROLOG的使用确实消除了属性-值语言的一些弊端。然而，在ILP中PROLOG的使用丢失了属性-值语言中的一个关键成分：类型的概念。在AVL中，每一个属性都隐含为一个类型，能取很多可能的值。通过抽取某一个属性(元组映射)并测试其值是否属于其相应的类型构造表达式(合取式)而建立样例与类之间的映射关系。而PROLOG没有类型系统，样例的所有特征都用谓词表达。结果使得目标映射的构造不受约束。为了使学习器能应用于实际领域，必须引入一些机制，如连接子句、模式声明、确定性等语法偏置。

为了解决属性-值和一阶逻辑二种知识表达方式在归纳学习中的弊端，人们提出了使用高阶逻辑作为ILP中的知识表达机制[2,3]，不仅能捕捉数据的类型信息，而且也能表达具有复杂结构的样例。在高阶逻辑的框架下，每一个样例都用一个封闭的项进行描述。这种知识表达方式不仅简洁——一个样例的所有信息都包含在一个位置，而且表达样例的项的结构能对目标映射的搜索提供一个有力的指导。能实现这种知识表达方式的就是Escher语言，它是一个类型化的(typed)、高阶的、集成了函数编程语言和逻辑编程语言优点的编程语言，它的语句是等式而不是PROLOG中的子句，计算模型采用重写机制[4,5]。

目前基于高阶逻辑的知识表达方式已经实现了基于规则的学习[2]、基于决策树的学习[6-11]和遗传算法[12-16]这三种学习算法，在解决一些现实问题上这些算法取得了不错的效果。

本文内容结构组织如下：首先归纳总结了复杂结构的归纳学习所产生的历史背景，然后论述了复杂结构归纳学习所

*)获国家自然科学基金项目(项目批准号：60675030)和《国家科技成果重点推广计划》项目(2003EC000001)资助。**李琳娜**　博士研究生，研究方向为数据挖掘；**杨炳儒**　教授，博士生导师，研究方向为知识发现与智能系统、柔性建模与集成技术；**周法国**　博士研究生，研究方向为自然语言处理。

使用的知识表示方式、复杂结构归纳学习的形式化定义及在高阶逻辑的知识表达方式下所采用的三种学习算法：基于规则的归纳学习、准确率驱动的决策树学习、遗传编程的学习。复杂结构的归纳学习在机器学习领域的应用及如何解决一些现实问题的讨论随后给出，最后分析了复杂结构的归纳学习的研究所面临的挑战性问题。

2 复杂结构归纳学习产生的历史背景

2.1 以属性-值与一阶逻辑作为知识表示方式的归纳学习

传统的归纳学习系统采用属性-值语言作为知识表达方式，即每一个样例用一个常量元组描述，元组的每一个分量是该样例的某个特征值。属性-值具有知识表达方式易于理解且采用其作为知识表达方式的学习器具有执行效率高的优点（由于直接操作样例的特征元组）。但在具有复杂数据结构类型的领域中，归纳学习如果使用属性-值知识表达方式，则存在如下弊端[3,7,17,18]：

1）首先，在处理复杂结构样例的归纳学习时，属性-值表达方式过于冗长。这是由于属性-值语言是基于命题逻辑的，是一种表达能力比较弱的形式化语言。

2）其次，由于样例通常具有很多特征，如何选择对归纳学习最有利的特征是一个很困难的问题，很难有一个对所有的应用领域都适用的特征选择标准。

3）再者，更为重要的是，在处理具有复杂结构的数据时，属性-值这种知识表达方式几乎不可能直接捕捉数据任意的结构信息。

在从复杂结构数据中归纳高层概念的时候，样例存在复杂的结构信息并且已被证明是必需的。例如，在化学领域，分子也许需要作为样例考虑，此处的分子就是具有自己特征的原子的复杂排列，可以用连接的原子作为一个图来表达分子；又如，将钥匙串上的一串钥匙表达为集合等。正如 Quinlan 在文献[18]所述："对象或观察样例的数据可能具有预定义的属性值不能捕捉的任意复杂结构"，这种情况在以属性-值学习中特别明显。

4）另外，不能够描述属性值之间的实质关系。

5）最后，对背景知识的利用方面具有一定的限制。

如何能够超越上述属性-值表达方式的局限，设计直接处理具有复杂结构的数据的归纳学习算法，使其以更加直接、自然、简洁的方式应用到复杂领域中，成为机器学习发展的方向与动力之一。

在这样的背景下，研究者提出了用一阶逻辑作为归纳学习系统中的知识表达语言，并由此产生了归纳逻辑编程（Inductive Logic Programming，ILP）这样一个研究领域，成功地开发了许多实用系统，如 FOIL，LINUS 等，这些系统都采用一阶逻辑语言 PROLOG 作为实现语言。基于一阶逻辑知识表示方式的 ILP 一定程度上解决了上述基于属性-值表达方式学习算法的问题，在复杂结构领域的归纳学习中取得了很多的成果。然而，仍存在以下几个方面的问题[2,3,5,7,8]：

1）首先，基于一阶逻辑 PROLOG 知识表示方式的 ILP 丢失了属性-值语言中的一个关键成分：类型的概念。一阶逻辑 PROLOG 知识表示方式是非类型化的，在实现技术上表现为扁平化的表达机制，即用一个事实数据库表达关于某个样例的信息。非类型化的扁平表达机制存在以下缺陷：

I. 没有类型信息并且包含在类型中的语义信息也都给丢失了。

II. 样例的信息分散在不同的位置，不利于学习过程中这些信息的使用。

III. 类型决定了能对样例进行的操作运算，这些操作运算在假设空间的构造过程中能有效地约束其规模，减少归纳假设的搜索空间。如果不使用任何约束机制，学习过程中不仅仅会生成语义上没有意义的谓词，而且会生成类型不正确的谓词。很多 ILP 研究认识到这一点，通过以语言偏置的形式（如模式声明、型声明等）实现有限的、对于具体应用有意义的特别指定的类型系统，缩小搜索空间和指导搜索过程。但是，语言偏置的添加一方面只对与其相适的目标定义的搜索有良好的效果，另一方面不适宜对于目标定义构造来说先验知识少的任务环境。因此扁平化的 ILP 知识表达机制缺乏统一、一般化的类型实现机制，以有效地构造搜索空间和设计搜索策略。

2）PROLOG 本质上是一阶的，不能提供高阶的构造器，也不能提供描述性元编程的灵活性。

3）基于一阶逻辑 PROLOG 知识表示方式的 ILP，为了减少空间搜索的复杂度，限制学习到的归纳定义中函数符号的出现，这在一定程度上限制了其应用于复杂结构的归纳学习领域。

4）已有研究显示，在一定的条件下，使用新谓词对成功的归纳学习是必需的。在基于一阶逻辑 PROLOG 知识表示方式的 ILP，谓词的发明和使用是一个尚未很好解决的问题，因为谓词的发明和使用是一个二阶的概念，只有在高阶环境中这个问题才可以很好地解决。

2.2 以高阶逻辑作为知识表示方式的归纳学习阶段

基于上述认识，归纳学习要全面地解决复杂结构数据领域的问题，明确地需要首先克服属性-值学习和基于一阶逻辑的 ILP 的内在缺陷。对这一方面的问题，机器学习与知识发现研究共同体内的研究者们从不同角度、不同层面进行了探索，并且逐渐认识到"类型化的高阶逻辑"，特别是类型化的高阶逻辑语言 Escher[4,5]，可以作为恰当的知识表达方式解决上述缺陷。本小节首先介绍达到这一认识的过程中相关的研究探讨，其次重点给出类型化的高阶逻辑语言 Escher 的一般特点。

提出类型化的高阶逻辑的知识表达方式之前，与之相关的思想与方法探索集中在如下工作中：

文献[19]较早地讨论了在一阶逻辑学习中用来控制算法复杂度的各种可能的知识表达方式，尤其是 RDT 系统所采用的知识表达方法。在文献[20]中描述了一个使用显式陈述性语言偏置的学习系统，这一语言偏置与高阶逻辑中所采用的重写规则具有相似的目的。文献[21]讨论了知识表达方式对学习系统计算复杂度的影响，比较分析了各种知识表达方式。在文献[22]中，研究了陈述逻辑的学习能力。陈述逻辑与高阶逻辑有一些相似之处，比如它们都有构造器，如 AND，ALL，AT-LEAST，它们与 Escher 所提供的表达式具有相似的语义。陈述逻辑的一个运算就是判断一个陈述是否比另一个陈述更有一般性，它最初应用在分类推理领域，在具体应用时，样例用原子描述；若它被目标概念所蕴含，则将它标记为正样例。文献[23]对 ILP 领域，陈述逻辑框架下的各种学习系统进行了比较。文献[24]建议学习系统使用高阶逻辑，但由于采用具有不完全高阶逻辑特征的 PROLOG 作为知识表达方式，故相应的学习系统有很大的限制。该文显示了作者的关于高阶逻辑作用的初步认识是正确的，但要达到其最终

目的需要用完全的高阶逻辑表达机制。文献[24]建议在机器学习中使用集合数据类型,并研究了在决策树学习中集合-值的特征。它只允许使用字符串集合,对系统的应用领域做了很多限制。整体看来,其方法和动机与在 ILP 中采用类型化的高阶逻辑的知识表达方式非常相似。文献[25,26]将决策树学习从属性-值表达方式扩展到表达能力更强的语言并开发了 Tilde 系统。在 Tilde 系统中,描述样例的项与高阶逻辑表达机制下描述样例的项不但在结构上相似,而且能处理相似的问题。但有几点显著不同:首先,Tilde 采用 PROLOG 的平坦表达方式;再者,它允许变量和条件跨越树的几个节点扩散,会导致许多复杂性问题。文献[26]中还介绍了几个优于 Tilde 的一阶逻辑框架下的决策树学习系统。文献[27,28]在学习系统中使用函数,采用函数逻辑编程系统(Functional Logic Programming System,FLIP)这一形式化机制,学习系统能处理多种学习任务,如学习函数的递归定义和数据挖掘任务。虽然它不具有显著的类型和高阶特征,但采用函数和等式理论的动机值得借鉴。

在上述工作的研究基础上,C. Giraud-Carrier,J. W. Lloyd 等人提出了类型化的高阶逻辑的知识表达方式。这方面最经典的知识表示系统是 Escher 语言。与基于命题逻辑和基于一阶逻辑的知识表达方式相比,其具有如下特征:

1) 可以设计直接处理具有复杂结构的数据的归纳学习算法,使其以更加直接、自然、简洁的方式应用到复杂领域中。

2) 支持各种数据类型,如集合、多集及图等任意复杂的类型,能描述复杂结构的样例。

3) 学习到的归纳定义具有统一的表达方式,并且更易于理解,能捕捉数据的本质特征。

4) 类型系统自然地包含类型信息及其语义信息。

5) 样例空间中的每一个样例都用一个封闭的项进行描述,将样例的所有信息集中在一个位置,有利于学习过程中这些信息的使用。

6) 类型系统能够阻止逻辑变量被无意义归一,限制变量的实例化。更重要的是,每一个类型都决定了可对其进行的操作,而这些操作被用来构造归纳假设,故其从一定程度上约束了假设语言,缩小了归纳假设的搜索空间。

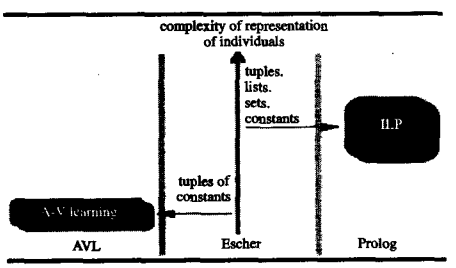

图 1 AVL,PROLOG,Escher 在表达能力方面的差别

7) 能以统一的方式处理函数和谓词。

我们将在下一节中详细介绍 Escher 语言。作为本节结论,这里通过图 1 形象地展示了 AVL,PROLOG,Escher 在表达能力方面的差别[3]。

从图 1 可以看出,属性-值语言仅仅能表达由常量组成的元组,PROLOG 能表达元组、链表、集合、常量等。而 Escher 能表达任意复杂的类型,它为属性-值学习和归纳逻辑编程提供了一个统一的框架。

3 类型化的高阶逻辑的知识表示语言 Escher 介绍

Escher 是一个强类型的描述性编程语言,集成了函数编程和逻辑编程语言的优点。以 Escher 作为知识表示方式,最主要的优势就在于相应的学习器能够直接归纳具有复杂内部结构的样例描述。Escher 能够表达如下类型以描述复杂结构样例:

• 整数、浮点数、字符串、字符集合、布尔型。这一组类型称为基本类型,是表达样例的项的基本构造块。在 Escher 中,Int 表示整数类型,Float 表示浮点类型,Char 表示字符串类型,String 表示字符串集合的类型,Bool 表示布尔类型。

• 数据构造器。对用户定义类型,数据构造器是必需的。0 元的数据构造器通常称为常数。数据构造器在 Escher 用 data 关键字声明。关键字 data 表示一个类型的声明以及该类型相应的数据构造器,每个数据构造器构造了一个以多个常数作为域值的类型,关键字 type 声明一个由其他类型构造的类型。

• 元组。元组本质上是属性-值表达方式中样例表达的基础,其作用显而易见。

• 集合、多元素集合。集合和多元素集合虽然没有元组类型应用范围广,但是集合尤其是多元素集合也是相当有用的数据类型。

• 链表。对链表可以进行的操作有:取链表的头元素、去除头元素后的尾链表及将链表类型转换为集合类型等。

• 树。树定义为 data Tree T= Node T[Tree T],其中 T 是节点的类型。

• 图。图又分为有向图和无向图。无向图定义为:

type Label = Int;

type Graph v w = {Label,v},{(Label→Int,w)}},其中 v 是顶点所表示的信息的类型,w 是边所表示的信息类型。有向图的定义类似。

下面举两个例子展示 Escher 的表达方式。全面详实的描述请参见文献[9]。

第一个例子是一个比较典型的归纳学习任务:根据天气情况判断是否打网球[29]。

首先 Escher 要给出样例的类型定义。在该例中样例 Weather 用元组类型表达,其具体定义如下:

data Outlook = Sunny | Overcast | Rain;

data Temperature = Hot | Mild | Cool;

data Humidity = High | Normal | Low;

data Wind = String | Medium | Weak;

type Weather = (Outlook, Temperature, Humidity, Wind),

上述声明定义了描述样例 Weather 的元组类型:该元组由 4 个属性组成,其类型分别为 Outlook,Temperature,Humidity,Wind。4 个属性类型由相应的 data 关键字分别定义。

需要学习的归纳假设具有如下形式:

playTennis::Weather→ Bool,

说明归纳学习的任务在于学习一个名为 playTennis 的函数,该函数的定义域类型是 Weather,值域为布尔型。

下面给出具体样例的描述:

playTennis (Overcast,Hot,High,Weak) =⊤

这是一个正例的描述,表示阴天、气温较高、空气湿度较大、风力比较弱的情况下可以打网球。

playTennis (Sunny, Hot, High, Weak) = ⊥

这是一个负例的描述，表示当天气晴朗、气温较高、空气湿度较大、风力比较弱的情况下不可以打网球。

第二个样例是另一个比较经典的学习问题——Mutagenesis 问题[30]：根据一个分子的结构预测其是否具有活性。最适合描述分子结构的数据类型是无向图，图中的节点表示原子，边表示原子之间的结合。首先定义化学元素的类型 Element：

data Element = Br|C|Cl|F|H|I|N|O|S;

同时也定义如下数据类型：

type AtomType = Nat;（用自然数表示原子类型。）

type Charge = Float;（用浮点数表示原子所带的电荷。）

type Atom = (Element, AtomType, Charge);（用元素、原子类型及原子所带电荷三个属性构成的元组表示原子。）

type Bond = Nat;（用自然数表示原子之间的结合信息）

基于上述类型声明可将分子定义为：

type Molecule = Graph Atom Bond;（用 Escher 内置的图类型表达分子，图的节点为 Atom 类型，图的边为 Bond 类型）

需要学习的归纳假设 active 形式如下：

activ:: Molecule → Bool

下面给出具体的样例描述例子，第一个是一个正样例，第二个是负样例：

active({(1,(C,22,−0.117)),(2,(O,45,−0.388)),(3,(C,22,−0.117)),(4,(C,195,−0.087)),(5,(C,195,0.013)),(6,(N,38,0.812)),(7,(O,40,−0.388)),(8,(O,40,−0.388)),(9,(H,3,0.412)),(10,(H,3,0.412)),(11,(C,27,−0.087)),(12,(C,27,0.013))},{(⟨1,2⟩,7),(⟨1,6⟩,7),(⟨2,3⟩,7),(⟨2,8⟩,1),(⟨3,4⟩,7),(⟨3,9⟩,1),(⟨4,5⟩,7),(⟨4,11⟩,7),(⟨5,6⟩,7),(⟨6,7⟩,2),(⟨6,8⟩,2),(⟨11,12⟩,7)}) =T

active({(1,(H,3,0.142)),(2,(C,22,−0.117)),(3,(C,22,−0.117)),(4,(C,195,−0.087)),(5,(C,195,0.013)),(6,(N,38,0.812)),(7,(O,40,−0.388)),(8,(O,40,−0.388)),(9,(H,3,0.142))},{(⟨1,2⟩,7),(⟨1,6⟩,7),(⟨1,7⟩,1),(⟨2,3⟩,7),(⟨2,8⟩,1),(⟨3,4⟩,7),(⟨3,9⟩,1),(⟨4,5⟩,7),(⟨5,6⟩,7),(⟨6,7⟩,3),(⟨6,8⟩,2)}) = ⊥

通过上面的两个例子已经了解到如何使用 Escher 描述归纳学习中的知识。下面根据归纳学习任务的一般结构，从样例、归纳假设和背景知识三个方面概括 Escher 在归纳学习中作为知识表示方式的特点。

3.1 样例的表示

Escher 在归纳学习中描述样例的基本原则为[9]：每一个样例个体都用一个封闭的项来描述，即所谓的“个体-项”的表达方式。在 Escher 中，一个样例用它的所有特征的集合来表示。描述项的语言及描述样例的语言完全由类型符号决定。关于样例的信息也就局部化在项中。子项命名在只有在其他子项中被引用时才需要。

这里需要指出的是，在基于一阶逻辑的 ILP 中，从解释中学习（Learning from Interpretation）方式[31]以解释方式描述样例，也能够实现样例信息的局部化。然而，用项的方式比以解释的方式更能对样例信息实现局部化。例如，人们很容易理解一个分子的这样描述：将一个原子及其与其它原子结合的所有信息都包含在一个子项中。虽然这种表达方式有可能带来冗余，但它将一个原子的所有信息限制在一个子项中。

3.2 背景知识的表示

类型系统能定义作用于不同类型上的操作，这些操作以函数的形式表示。从归纳学习的角度看，函数描述了不同层次的背景知识。背景知识的意义在于，一方面详细描述了样例的相关结构信息，另一方面从归纳的角度看，为假设函数的构建提供基本的可用信息，因而假设的最后构造是由定义在类型上的函数形成的，这在下一小节关于假设表示的例子中可见。

为了便于理解和分析，不同的函数操作或背景知识可以分为三个层面：

首先，从复杂的类型中抽取子项的选择函数，每一个复杂的类型都有其适用的选择函数。例如，投射函数适用于元组类型；集合成员函数适用于集合类型；链表成员函数适用于链表类型等。没有选择函数，学习器就无法使用类型的内部结构。

其次，普通的、独立于具体应用的函数，比如标准的集合和链表运算，这些函数也与归纳定义中的类型紧紧相关联。区别在于选择函数自动包含在假设语言中。然而，其它的背景函数可以不被包含在假设语言中。在 Escher 中，选择函数和一般函数都可以由独立的模块提供，不需要由用户定义。

最后，存在独立于具体的应用领域且很难被学习器自身发现的辅助函数，这些函数表达了更为面向具体应用的背景知识，它们的定义必须由用户提供。

上述三种函数的具体定义可以通过如下统一的方式实现。它们可以是针对每个数据类型定义的变换，一个变换就是一个基本函数；也可以是由基本函数通过合成运算得到的函数。首先看一下变换的形式化定义。

一个变换定义为：$(R_1 \to \text{Bool}) \to \cdots \to (R_K \to \text{Bool}) \to S \to T$，这里，$R_1, \cdots, R_K, S, T$ 都是类型，且 $K \geqslant 0$，$R_1, \cdots, R_K$ 和 T 的任何参数都在 S 中出现，S 称为变换的源，T 称为变换的目标，K 是变换的阶。它表示若 $r_i :: R_i \to \text{Bool}$，$i = 1, \cdots, k$，都是函数，那么 $(f\ r_1 \cdots\ r_k)$ 是类型为 $S \to T$ 的函数。

其次看一下函数的合成运算的定义。用‘·’表示合成函数的运算，定义为：$(R \to S) \to (S \to T) \to (R \to T)$，它表示 $(f.g)x = g(fx)$，这里 R, S, T 都是类型。

下面举例论述变换和函数的具体表示。

对上述打网球的例子，包含如下变换：

projOutlook:: Weather → Outlook

它表示对关于天气的元组在 Outlook 分量上做投影运算。

(= =Sunny):: Outlook→Bool

它判断某个 Outlook 类型的值是否等于 Sunny。

合成函数(projOutlook · (= =Sunny))

是对上述两个变换的合成，作用对象的数据类型是 Weather，它表示先对某个天气元组在 Outlook 分量上做投影运算，然后判断投影运算的结果是否等于 Sunny。

表达式($\wedge_2\ q\ p$)

表示对谓词 p 与 q 做与运算。

3.3 假设的表示

给定一个类型标记 $f :: X \to Y$，需要学习的 Escher 假设定义具有 $f(x) =$ if E then s else t 的形式。这里 x 是类型为 X 的变量，s 和 t 可以是类型为 Y 的值，也可以是 if-then-else 表

达式，E 是一个布尔表达式。这是一个非常一般的形式，但需要限制 Y 是一个有限元素集合。通常情况下，Y 也确实如此。学习这种形式的函数定义意味着需要实例化 E，s 和 t。

下面我们结合具体例子展示上述一般假设形式的具体实现。在前述打网球的例子中，可学习到 playTennis 函数的定义如下：

```
playTennis ω= if Λ2(projOutlook • ( = = Sunny))
                (projHumidity • ( = =High)) ω
              then ⊥
              else if Λ2(projOutlook • ( = = Rain))
                (projWind • (= = Strong)) ω
              then ⊥
              else ⊤
```

它表示当天气晴朗、空气湿度高的情况下或者下雨、风力较强的情况下不可以打网球，其它的情况下都可以打网球。

通常，自顶向下的精化方法需要考虑整个搜索空间的很大一部分。例如，PROLOG 的一个平凡实现的精化运算可以生成包含所有变量的文字。进一步说，在精化运算过程中的任一阶段，需要考虑所有可用的文字。然而，无用的文字因为不能再生成任何有用的信息，在评估阶段被剔除掉。在前一阶段，由精化运算生成它们是浪费的。而类型系统的每一个数据类型都决定了可对其进行的运算，这在假设空间的构造过程中避免了无用文字的引入。Escher 表示的归纳假设是连接的，归纳假设的连接性质可以用类似如下的定义予以实现：除非 $b1$ 的类型由规则的前面部分决定，否则不能引进新文字 bondType(b_1)= =7。这样，非连接子句的概念可以看作是非类型化语言的范畴，而不是 Escher 这样的类型化语言的范畴。相似地，如基于一阶逻辑的 ILP 中的模式声明等语法偏置可以由一个合适的类型系统替代。由此可见，类型系统是一个表达和使用陈述性偏置的强有力的工具。

在复杂结构的归纳学习中，类型及与其相关的函数和存在变量及其数量共同约束假设语言，从而决定了归纳学习过程的复杂程度。而在属性-值学习中，由于不允许使用存在变量，故称假设语言完全由类型决定。由此可见，不同知识表示机制下的学习任务之间的差别在一定程度上是由它们所采用的形式化描述机制的差别决定的(如图 1 所示)。

4 基于复杂结构的归纳学习的一般化定义

假设有一个依赖于具体应用的字母表，足够包含下列成分中的所有符号，那么复杂结构的归纳学习可以定义如下[8]：

给定：

1)目标函数的标记

目标函数 $f:\sigma\rightarrow\tau$ 是需要从样例中学习其定义的函数，假定 f 不出现在背景理论的字母表中。这里，σ 是域的类型，它可以是任意复杂的类型，也可以是高阶的；τ 是 f 的值域的类型，一般仅仅由几个元素构成，通常情况下，是一个布尔类型。

2) 样例

每一个样例具有 $f(s)=t$ 的形式，这里 s 跟 t 分别是具有类型 σ、τ 的封闭项，在很多情况下 s 是值的元组，t 是一个值。

3) 假设语言

用来构造目标函数定义的语言。

4) 背景理论

由出现在假设语言中的函数的定义构成，它是对学习算法可用的知识，用一个一致的程序描述。

5) 覆盖关系

若项 $f(s)=t$ 是 f 的归纳定义和背景理论的逻辑后承，则称目标函数 f 的一个定义覆盖了样例 $f(s)=t$。

发现：

在假设语言中的目标函数的定义，并且

1) 它覆盖所有正样例；

2) 泛化(能准确地应用于新的样例)。

总而言之，归纳学习就是搜索能覆盖所有样例并且满足背景理论的目标函数的定义。

归纳学习问题本质上也是一个启发式搜索问题。可能有很多目标函数的定义，搜索空间就是由这些定义构成的。找到比较好的启发式信息将搜索空间极小化是一个重要的研究方向。

基于第 3 节的知识表示方式，基于高阶逻辑的复杂结构归纳学习实质上就是，在“个体-项”描述下，搜索适宜的目标函数定义。为了达到这样一个目的，必须探究个体描述项的内在结构，通过这些结构特征形成判断谓词，个体是否满足该谓词成为目标函数具体取值的标准。从而，从搜索的角度看，目标函数的搜索过程实质是一个谓词构建的过程。项内部结构特征的描述通过以背景知识形式体现的定义在类型上的函数获得，因而搜索空间中的谓词是由基本变换获得的函数通过合成运算得到的。此处需要指出的是，搜索空间中的谓词都必须是标准谓词。

为了说明重写的概念，给出谓词之间蕴涵关系的定义。

定义 1 设 p，q 都是标准谓词，若 q 与 p 具有相同的类型且 $\forall x.((p\,x)\leftarrow(q\,x))$ 是背景知识 B 的逻辑后承，则称谓词 q 蕴含谓词 p，记为 $p\Leftarrow q$。

每一个重写(rewrite)具有 $r\Leftarrow s$ 的形式，r 和 s 是具有相同类型的谓词且 $r\Leftarrow s$。

若 r 是谓词 p 中的一个子项，则能将重写 $r\Leftarrow s$ 作用于谓词 p，得到新的谓词 $p[r/s]$。此处需要注意的是，将重写 $r\Leftarrow s$ 作用于谓词 p 的时候，必须确保得到的新谓词 $p[r/s]$ 蕴含谓词 p，即只能将重写作用于谓词 p 的“正确”子项。关于这一要求的具体技术细节见参考文献[32,33]。

对任何数据类型，都有一个变换 top::$T\rightarrow$Bool；定义为 top x = True；Top 是一个返回值总为 True 的谓词，任意假设语言都包含 top，它可作用于任意的数据类型。

搜索空间的构造首先从最弱的谓词 top 开始，然后将所有头部匹配 top 的重写策略作用于 top 得到新的谓词，继而通过对已得到的谓词不停地应用重写策略枚举谓词[34]。

谓词空间的搜索有可能不是一棵树，而是一个图。为了避免同一个谓词被重复测试，记录所有已测试过的所有谓词(是一个集合，可以存放在 hash 表中)。当生成一个新的谓词的时候，检查它是否已被测试过。

为了解决语法上不同的等价谓词被重复搜索的问题，可以在谓词上定义一个全序关系。

5 目前已存在的复杂结构归纳学习算法

目前已存在的复杂结构的归纳学习算法主要分为基于规则的学习、准确率驱动的决策树学习和遗传算法。下面分别论述。

5.1 基于规则的学习

基于规则的学习算法将学习到的假设表示为 if-then 规则的集合，使用序列覆盖的思想，一次学习一个规则，以递增的方式形成最终的规则集合。由于采用 Escher 语言作为知

识的表达机制，学习到的每一条规则都是 Escher 语句。由于 Escher 基于等式理论，Escher 中的语句都是用等式的形式表达，等式左边称为语句的头，等式右边称为语句的体。一般来说，目标函数由一个或多个语句组成，最常见的形式是仅包含一个语句、头部具有最一般形式的非递归定义。

一阶逻辑表达机制下的 ILP 最初应用领域是程序合成，高阶逻辑表达机制下的 ILP 也不例外。下面详细论述后者在程序合成领域的基于规则的归纳学习过程。

对程序合成而言，归纳定义通常包含不止一个语句(定义多是递归的)。这是由于目标函数通常对具有几个数据构造器的数据类型运算，对每一个数据构造器都有一个函数的定义，归纳到的目标函数是由这些定义共同构成的。每一个归纳定义具有如下形式：

$f(r)=$ if E_1 then t_1 else
　　if E_2 then t_2 else
　　…
　　if E_n then t_n else t_0

这里，每个 E_i 都是一个公式，其自由变量都必须在 $f(r)$ 中出现。用来构造每一个 E_i 的字母必须出现在背景理论中，也有可能包含符号 f。若包含了符号 f 归纳定义是递归的，否则是非递归的。$t_1,\cdots,t_n$ 是所有样例等式右边的值，t_0 不一定在某个样例等式右边出现，但必须是类型为 τ 的值。

算法首先选择一个参数，其每一个数据构造器对应目标函数的一个定义，根据选择参数的数据构造器的不同将样例分类。若有一类样例称为 R，对 R 中的样例根据等式右边值的不同将其分为 $(n+1)$(有可能小于)个类，每一类等式右边的值为 t_i。对每一个 t_i 对应的样例构造公式 E_i，E_i 需要覆盖 t_i 中的所有样例。将每个 t_i 对应的样例类处理完毕就得到一个归纳定义。其它的数据构造器对应的归纳定义学习过程相同，在此不再详细论述。

利用标准的 A* 算法搜索每个表达式 E_i。搜索空间中的每个节点都对应于一个公式。开始节点是公式 True，目标节点是满足上面所陈述条件的任意公式。将各种运算作用于一个公式，从而对其进行扩展，得到需要评估的后继公式。选择对哪一个公式扩展，是由一个启发式函数指导的。启发式信息主要由样例提供。若一个公式确实覆盖了它应该覆盖的样例且不满足它不应该满足的很多样例，就意味着这个公式对应的节点很有可能是目标节点路径上的节点。

算法 1 是该算法的一般框架。

算法 1　基于规则的算法

输入：背景理论 B，标记 $\sigma\rightarrow\tau$
输出：目标函数 f 的定义 D
1. $D_0=\phi$;
2. 选择一个类型为 τ 的缺省值 t_0;
3. V=所有类型为 τ 的非缺省值的集合;
4. 选择 D 中语句的头部分的参数 $r_1,\cdots,r_p$ $(p\geqslant 1)$
5. for j=1,…,p do{
6. $R_j=\varepsilon$ 中对应于参数 r_j 的样例集合;
7. 对某个 s，$D_j=R_j$ 中形式为 $f(s)=t_0$ 的样例集合;
8. $c_j=\phi$;
9. i=0;
10. while $R_j\backslash D_j\neq\phi$ do{
11. 　　i=i+1;
12. 对某个 s，搜索 t_i，公式 E_i 及非空集合 $N_{ji}\subseteq R_j\backslash D_j$，满足：
　(i)　对所有的 $f(s)=t\in C_j\cup N_{ji}$，f(s)根据程序 P 归纳到 t;
　(ii)　对所有的 $f(s)=t_0\in R_j\backslash N_{ji}$，f(s)根据程序 P 归纳到 t_0;
　其中，$P=B\cup D_{j-1}\cup\{f(r_j)=$ if E_1 then t_1 else…
　　if E_i the t_i else $t_0\}$;
13. 　　}
14. $D_j=D_{j-1}\cup\{f(r_j)\}=$ if E_1 then t_1 else…
　　if E_i then t_i else $t_0\}$;
15. D=Dp;
16. 返回 D;

5.2　基于决策树的复杂结构归纳学习

基于决策树的复杂结构归纳学习采用机器学习领域比较经典的决策树学习的思想，样例划分的标准是谓词，采用自顶向下、逐步精化的归纳学习过程。

在具体讨论决策树算法之前，先给出一些关于准确率启发式信息方面的定义。

定义 2　ε 是共有 c 个类的所有样例的集合，设第 $i(i=1,\cdots,c)$ 个类中有 n_i 个样例，$j=\text{argmax}\{n_i\}$，则第 j 类称为最大的类。

定义 3　ε 是共有 c 个类的所有样例的集合，N 是 ε 中的样例的个数，n_i 是 ε 中第 i 个类中的样例个数且定义 $p_i=n_i/N(i=1,\cdots,c)$，则 ε 的准确率定义为 $A_\varepsilon=p_M$，M 是 ε 中最大的类的索引。

定义 4　若 $P=\{\varepsilon_1,\cdots,\varepsilon_m\}$ 是 N 个样例的一个划分，ε_j 中有 N_j 个样例，$j=1,\cdots,m$，那么 P 的划分准确率定义为：

$$A_P\equiv\sum_{j=1}^{m}\frac{N_j}{N}A_{\varepsilon_j}$$

定义 5　ε 是样例集，$\{\varepsilon_1,\varepsilon_2\}$ 是 ε 的一个划分，$\{\varepsilon_1',\varepsilon_2'\}$ 是 ε 的另一个不同的划分，若 $\varepsilon_1'\subseteq\varepsilon_1$，则称 $\{\varepsilon_1',\varepsilon_2'\}$ 是 $\{\varepsilon_1,\varepsilon_2\}$ 的精化。

定义 6　$P=\{\varepsilon_1,\varepsilon_2\}$ 是 N 个样例集合 ε 的一个划分，n_i 是 ε 中第 i 个类中样例的个数，$n_{j,i}$ 是第 i 个类中在 ε_j 中的样例个数，$j=1,2$，$i=1,\cdots,c$，则划分 P 的精化界 B_P 定义为

$$B_P\equiv(\max_i\{n_i+\max_{k\neq i}n_{1,k}\})/N$$

定理 1　ε 是样例的集合，P 是 ε 的一个二元划分，若 P' 是 P 的一个精化，则 $A_{P'}\leqslant B_P$。

当搜索划分一个节点的谓词时，该定理用来控制对一些无用谓词的搜索。在搜索过程中，学习系统记录当前最好的划分 P 及其准确率 A_P。当测试一个新的划分 Q 时，若 $B_Q\leqslant A_P$，则划分 Q 及其所有的精化都可以被剪枝。

决策树算法从具有一个且仅一个叶子节点的决策树开始，不停地选择具有最低准确率的叶子节点 n(根据准确率启发式信息)进行扩展，在节点 n 搜索谓词 p 使其满足：p 能精化节点 n 对应的划分且 p 是节点 n 对应的整个谓词空间中最高准确率划分的谓词，然后将节点 n 扩展为两个新的叶子节点，一个分支满足谓词 p，另一个分支不满足谓词 p。若节点 n 对应划分的准确率不能再提高或者决策树中节点的个数超过某个确定的上界，算法终止。在决策树的构造过程中，每一个分支用谓词标记。算法结束时，每一个叶子节点用它最大的类标记。算法描述如算法 2 所示。

算法 2　决策树算法

输入：ε：样例集合
　　S：决策树中节点个数的上界
输出：决策树 tree
1. tree：=具有一个节点的树;
2. finished：=false;
3. while not finished do
4. n：=tree 的具有最小划分准确率的叶子节点;
5. F：=与节点 n 相关的所有样例的集合;
6. p：=搜索能对 n 对应的划分产生具有最高划分准确率精化的谓词 p;
7. P：=谓词 p 对 F 的划分;
8. If $A_P>A_F$
9. then tree：=将 tree 的节点 n 扩展为两个节点，一个分支满足 p，另一个分枝满足 p 的否定;
10. S：=S−1;
11. finished：=(s=0);
12. Else
13. Finished：=true;

5.3　基于遗传编程的复杂结构归纳学习

遗传算法是一种大致基于模拟进化的学习方法，其中的

假设常被描述为二进制位串，位串的含义依赖于具体的应用。然而，假设也可以被描述为符号表达式甚至是计算机程序。搜索合适的假设是从若干初始假设的群体或集合开始的。当前全体的成员通过模仿生物进化的方法来产生下一代群体，比如说随机变异和交叉。每一步，根据给定的适应度评估当前群体中的假设，而后使用概率方法选出适应度最高的假设作为产生下一代的种子。用位串描述假设称为遗传算法(Generic Algorithm,GA)，用计算机程序描述假设称为遗传编程(Generic Program-ming,GP)[29]。

遗传编程技术已经成功应用到命题逻辑[35]和一阶逻辑环境下的归纳学习[36]。文献[14]讨论了进化高阶归纳学习的思想和基本假设。文献[15]详细描述了强类型进化编程系统(STEPS)的实现及其应用，该系统是采用类型化的高阶逻辑描述机制的归纳学习系统。在 STEPS 中，仍采用 Escher 表达知识，每个样例用封闭的项描述，通过选择操作可以抽取项的样例成分。项所使用的每一个结构都有它自己相关的选择操作运算集合，但归纳定义和选择运算都用程序树的形式描述。

由于 Escher 是一个强类型语言，进化范例必须具有类型信息，在学习的过程中只能生成的程序必须具有正确的类型。传统的进化编程技术(GP)规定程序树中程序体的所有函数是封闭的，即函数集合中的每一个函数只可以从该集合中的其它函数返回的值及数据类型中取值。这一约束虽然简化了遗传运算，但也限制了其应用。将类型系统引入到标准的 GP 得到的强类型遗传编程(STGP)(Strongly Typed Genetic Program-ming,STGP)[13]通过只考虑正确类型的程序来约束搜索空间克服了该问题。STEPS 通过允许高效地搜索描述能力更强的 Escher 归纳假设扩展了 STGP。下面具体阐述 STEPS 中初始种群的生成、交叉运算和变异运算的实现：

1) 通过从问题字母表中随机选择子树得到初始树群体，子树中的每一个节点有一个类型符号指明其参数类型及返回值类型。用来填充某一个空白槽的子树必须具有合适的类型，填充空白槽的局部变量必须在量化范围内(即类型和变量一致性约束)。

2) 交叉：在程序的进化过程中只保持类型和变量一致的程序树使得只有语法正确的程序才会进化。除此之外，必须保持选择函数子树的结构。这导致了交叉只能应用到程序树的某些节点。这些交叉点对应于函数集合中子树的根。一旦随机选择某个交叉点作为第一个双亲，第二个双亲必须满足保持类型和变量一致性约束。若没有这样的交叉点，那么从第一个双亲中选择一个交叉点。重复此过程。

3) 随机变异：随着进化进程的迭代进行，群体中的遗传变异的个数减小。极端的情况下，能导致产生较优解决方案的遗传信息的丢失。STEPS 采用 6 种不同形式的变异防止此问题的出现，从而保持遗传的多样性。

另外，STGPS 利用锦标赛选择技术作为假设的选择方法，用程序树在样例集上的预测准确率评估适应度。它从一个随机生成的、初始的程序树种群出发，重复地使用遗传运算，直到找到一个较优的解决方案。

程序树的深度决定遗传运算的选择。如果树的深度大于预定义的最大深度，则只能选用删除合取项或析取项的变异运算；若树的深度小于预定义的最小深度，则只能选用增加合取项或析取项的变异运算智能。若一个程序树的后代早已在群体中存在，需要对该树做变异运算。文献[16]中详细论述了强类型进化编程的深度控制策略。

6 复杂结构归纳学习的应用

复杂结构归纳学习在机器学习领域的典型任务和其它学科领域中皆有良好的应用，本节简要介绍其在元学习任务与分子生物学中的应用。

6.1 在元学习中的应用

元学习处理模型选择问题，是归纳从任务到学习器的映射。目前很多的元学习采取首先抽取任务的特征，然后归纳特征到合适的学习模型的映射的学习机制。可将抽取特征的方法分为统计和信息理论特征、标记和基于决策树的特征三类。然而，这些方法都是手工或预先计算一些特征。文献[37]提出了直接从归纳的决策树中学习，即复杂结构归纳学习应用于元学习问题。首先用一定的决策树学习算法归纳出任务所对应的决策树，将树中每个节点的信息用一个元组进行表示。元组包含的具体信息视实际情况而定，可以是该节点所使用的属性、所包含的样例的个数及一些启发式信息，然后将决策树转化为 Escher 所对应的封闭项，用来刻画学习任务的特征。然后对这些得到的 Escher 项应用第 3 节所述的学习算法归纳得到元理论，其思想描述如下：

1)选择一个数据集的集合 D。

2)对每一个数据集 $d\in D$，利用标准的决策树学习器(如 C4.5)归纳一个决策树，从而将 D 转化为决策树的集合 T。

3)对每一个 $t\in T$，构造相应的 Escher 项并对之做标记，从而得到一个元学习的样例。

4)对上一步得到的样例应用第三部分所述的学习算法，归纳得到元理论。

6.2 在分子生物学中的应用

20 世纪 90 年代中后期，T. G. Dietterich 等人[38]对药物活性预测问题进行了研究。其目的是让学习系统通过对已知适宜或不适宜制药的分子进行分析，以便尽可能正确地预测某种新的分子是否适合制造药物。该问题的困难主要在于，每个分子都有很多种可能的低能形状，而生物化学专家目前只知道哪些分子适用于制药，并不知道具体的哪一种形状起到了决定性作用。如果直接使用监督学习框架，将适合制药的分子的所有低能形状都作为正例，而将所有不适合制药的分子的所有低能形状都作为反例，则会由于正例中噪音度太高而难以成功地进行学习。这是因为一个分子可能有上百种低能形状，而这么多形状中只要有一种是合适的，这个分子就适合制药。为了解决这个问题，T. G. Dietterich 等人将每一个分子作为一个包，分子的每一种低能形状作为包中的一个示例，由此提出了多示例学习的概念。

麝香分子数据集是多实例问题学习中一个比较有名的数据集，将本文第 3 节的学习算法作用于该数据集，证实它们优于很多其它的学习算法，但稍微劣于专门针对此特定问题开发的学习器。预测一个分子是否产生了变异是多实例学习领域另一个比较有名的问题。将本文第 3 节的学习算法作用于解决此问题也取得了不错的效果。

结束语 基于高阶逻辑的复杂结构归纳学习克服了传统的属性-值学习与基于一阶逻辑的 ILP 技术的缺陷。类型化的高阶逻辑作为知识表达方式，提供了表达能力较强的假设语言，支持各种复杂数据类型，如集合、多元素集合(multi-sets)及图等。在这样的假设语言下得到的归纳定义易于理解，能捕捉数据的本质特征。

增加的表达能力也许会导致搜索空间的爆炸，从而影响学习系统的效率。然而，算法所需的额外代价可能小于为了使用传统的学习算法所需的隐含代价。进一步说，一些应用领域，如分子生物学，重点是获得数据的深层结构，归纳的知识的准确率和可理解性的度量比执行时间重要得多。最后，搜索过程中的额外的复杂度仅仅反映应用领域的内部结构的复杂程度。如果样例具有复杂的结构，那么不可避免地存在大量的特征可能被用来对样例进行分类。

基于类型化的高阶逻辑作为复杂结构归纳学习的知识表达方式是一个比较新的研究领域，国内外学术界对这一领域的研究并不多见。随着机器学习与知识发现在复杂结构领域应用深度和广度的拓展，如计算生物学、医学、病毒营销、反恐、语义 Web、社会网络分析、普适计算等领域的应用，我们相信基于高阶逻辑的复杂结构归纳学习必将有着广阔的发展前景。

最后我们指出复杂结构数据的归纳学习研究中所需要解决的重点问题：

1) 学习系统效率与可扩展性的提高，使现有学习算法能适用于现实中的大规模问题。

2) 背景知识中重写规则集的自动提供，而不是由人工提供。

3) 使决策树学习算法不仅仅能处理分类问题，还能用来处理回归问题。

4) 将类型化的高阶逻辑的知识表达方式应用到机器学习的各个领域，包括将类型化的高阶逻辑应用到进化学习领域、贝叶斯网络和强化学习中，将现有算法与核方法结合起来等。

5) 类型化的高阶逻辑的知识表达方式对计算理论的影响。

参 考 文 献

[1] Muggleton S, Raedt L D. Inductive Logic Programming: Theory and Methods. Journal of Logic Programming, 1994, 20 (19): 629-679

[2] Bowers A F, Giraud-Carrier C, Kennedy C, et al. A framework for Higher-order Inductive Machine Learning // Representation Issues in Reasoning and Learning. Area Meeting of Compulog-Net Area Computational Logic and Machine Learning, 1997

[3] Flach P A, Giraud-Carrier C, Lloyd J W. Strongly Typed Inductive Concept Learning // Proceedings of the Eighth International Conference on Inductive Logic Programming. LNAI 1446, Springer-Verlag, 1998: 185-194

[4] Lloyd J W. Declarative Programming in Escher. Technical Report CSTR-95-013. Department of Computer Science. University of Bristol, 1995

[5] Lloyd J W. Programming in an Integrated Functional and Logic Language. Journal of Functional and Logic Programming, 1999 (3)

[6] Bowers A F. Early Experiments with a Higher-order Decision-tree Learner // Proceedings of the COMPULOGNet Area Meeting on Computational Logic and Machine Learning. 1998: 42-48

[7] Bowers A F, Giraud-Carrier C, Lloyd J W. A Unifying View of Knowledge Representation for Inductive Learning. Available at: http://users. rsise. anu. edu. au/~jwl/, 2000

[8] Bowers A F, Giraud-Carrier C, Lloyd J W. Classification of Individuals with Complex Structure // Langley P, ed. Machine Learning: Proceedings of the Seventeenth International Conference (ICML2000). Morgan Kaufmann, 2000: 81-88

[9] Bowers A F, Giraud-Carrier C, Lloyd J W. A Knowledge Representation Frame-work for Inductive Leanring. Available at: http://csl. anu. edu. au/~jwl. 2001

[10] Lloyd J W. Knowldege Representation, Computation and Learning in Higher-order Logic. Available at: http://csl. anu. edu. au/~jwl, 2001

[11] Ng K S, Lloyd J W. Predicate Selection for Structured Decision Trees. ILP, 2005

[12] Montana D J. Strongly Typed Genetic Programming. Evolutionary Computation, 1995, 3(2): 199-230

[13] Kennedy C J. Evolutionary Higher-order Concept Learning // Koza J R, ed. Late Breaking Papers at the Genetic Programming 1998 Conference. University of Wisconsin, Madison, Wisconsin, USA, July 1998: 22-25

[14] Kennedy C J, Giraud-Carrier C. An Evolutionary Approach to Concept Learning with Structured Data // Proceedings of the Fourth International Conference on Artificial Neural Networks and Genetic Algorithms. 1999: 331-336

[15] Kennedy D J, Claire J, Giraud-Carrier C. A Depth Controlling Strategy for Strongly Typed Evolutionary Programming // Proceedings of the Genetic and Evolution -ary Computation Conference. Vol. 1. Morgan Kaufmann, Orlando, Florida, USA, 1999: 879-885

[16] Kenney C J. Strongly Typed Evolutionary Programming. PhD Thesis. University of Bristol, 2000

[17] Quinlan J R. Learning First-order Definitions of Functions. Journal of Artificial Intelligence Research, 1996, 5: 139-161

[18] Flach P A. The use of functional and logic languages in machine learning. Available at: http:// www. cs. bris. ac. uk/~flach/. 2000

[19] Kietz J-U, Wrobel S. Controlling the Complexity of Learning in Logic through Syntactic and Task-oriented Models // Muggleton S, ed. Inductive Logic Programming. chapter 16, Academic Press, 1992: 335-359

[20] Cohen W W. Rapid Propotyping of ILP Systems Using Explicit Bias // Proceedings of the IJCAI'93 Workshop on Inductive Logic Programming. 1993

[21] Neri F, Saitta L. Knowledge Represen -tation in Machine Learning // Bergadano F, De Raedt L, eds. Proceedings of the European Conference on Machine Learning. LNAI 784. Springer-Verlag, 1994: 20-27

[22] Cohen W W, Hirsh H. Learning the CLASSIC Description Logic: Theoretical and Experimental Results // Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning. Morgan Kaufmann, 1994: 121-133

[23] Muggleton S, Page C D. Beyond First-order Llearning: Inductive Learning with Higher-order Logic. Technical Report PRG-TR-13-94. Oxford University Computing Laboratory, 1994

[24] Cohen W W. Learning Ttrees and Trules with Set-valued Features // Proceedings of the Thirteenth National Conference on Artificial Intelligence. Menlo Park, CA AAAI Press, 1996: 709-716

[25] Blockeel H, Raedt L D. Top-down Inductive of Logical Decision Trees. Available at: http://www. cs. kuleuven. ac. be/ ~hendrik, 1997

[26] Blockeel H, Raedt L D. Top-down Inductive of First-order Logical Decision Trees. Artificial Intelligence, 1998, 101: 285-297

[27] Ferri-Ramírez C, Hernández-Orallo J, Ramírez-Quintana M J. Learning Functional Logic Classification Concepts from Databases // Proceedings of the 9th International Workshop on Funcitonal and Logic Programming (WFLP 2000). UPV University Press, Valencia, 2000(2039), 2000: 296-308

[28] Ferri-Ramírez C, Hernández-Orallo J, Ramírez-Quintana M J. Incremental Learning of Functional Logic Programs // Proceedings of the 5th International Symposium on Funcitonal and Logic Programming (FLOPS 2001). Springer, 2001: 233-247

[29] Mitchell T M. Machine Learning. McGraw-Hill, 1997

[30] King R D, Muggleton S, Srinivasan A, et al. Structure-activity Relationships Derived by Machine Learning: The Use of Atoms and Their Bond Connectivities to Predict Mutagenicity by Inductive Logic Programming // Proceedings of the National Academy of Sciences. 1996, 93: 438-442

[31] De Raedt L, Dehaspe L. Clausal Discovery. Machine Learning, 1997, 26(2/3): 99-146

[32] Lloyd J W. Higher-order Computational Logic // Kakas A, Sadri F, eds. Computational Logic: From Logic Programming into the Future. Springer-Verlag, 2001

[33] Lloyd J W. Learning Comprehensible Theories from Structured Data. 2003 (2600). Lecture Notes in Computer Science, Springer, 2003

[34] Lloyd J W. Predicate Construction in Higher-order Logic. Electronic Transactions on Artificial Intelligence, Section B, 2000, 4: 21-51

[35] Koza J R. Concept Formation and Decision Tree Induction Using the Genetic Programming Paradigm // Schwefel H-P, Männer R, eds. Parallel Problem Solving from Nature. 1990: 124-128

[36] Wong M L, Leung K S. Genetic Logic Programming and Application. IEEE Expert, October 1995

[37] Bensusan H, Giraud-Carrier C, Kennedy C. A Higher-order Approach to Meta-learning // Proceedings of the ECML' 2000 Workshop on Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination. ECML' 2000. June 2000: 109-117

[38] Dietterich T G, Lathrop R H, Lozano -Pérez T. Solving the Multiple Instance Problem with Axis-parallel Rectangles. Artificial Intelligence, 1997, 89: 31-71