# 专家系统外壳Pro/3

# 工程与实践

崔奇明 胡绍刚 管祖元 梁 凯
刘 君 王 玲 崔舒婷 胡 畔 编著

東北大學出版社
Northeastern University Press

# 专家系统外壳 Pro/3 工程与实践

崔奇明　胡绍刚　管祖元　梁　凯
刘　君　王　玲　崔舒婷　胡　畔　编著

东北大学出版社
·沈　阳·

**图书在版编目（CIP）数据**

专家系统外壳 Pro/3 工程与实践 / 崔奇明等编著. — 沈阳：东北大学出版社，2013.10
ISBN 978-7-5517-0455-7

Ⅰ.①专… Ⅱ.①崔… Ⅲ.①专家系统 Ⅳ.①TP182

中国版本图书馆 CIP 数据核字（2013）第 242312 号

**内 容 简 介**

本书系统地介绍了产生式系统类型的专家系统外壳 Pro/3 及其应用，并给出了较多的示例。全书共分 15 章：第 1 章 Pro/3 概述，第 2 章 Pro/3 基本组成，第 3 章 Pro/3 安装与开始，第 4 章事实，第 5 章规则，第 6 章非精确规则，第 7 章语法，第 8 章查询，第 9 章知识库，第 10 章推理引擎，第 11 章知识共享和交换，第 12 章 Pro/3 部分样本知识库，第 13 章 The Einstein Puzzle（爱因斯坦谜题）与 Pro/3 问题域模型化过程，第 14 章 Pro/3 与 MS Visio 的结合应用及文件扫描实用程序 FileScanner、第 15 章 Pro/3 应用研究及规则定义过程示例。

本书取材新颖、内容丰富、图文并茂，既注重理论又面向应用。本书是作者根据多年从事专家系统应用研究及对 Pro/3 系统的学习、翻译及整理、总结及其应用而编写的。本书需要读者具有专家系统基础知识，可作为高等院校计算机、自动化、信息管理等相关专业本科生或研究生关于专家系统及应用的教学参考书，也可供其他专业的师生及科研和工程技术人员自学或参考。

出 版 者：东北大学出版社
地址：沈阳市和平区文化路 3 号巷 11 号
邮编：110004
电话：024—83687331（市场部）　83680267（社务室）
传真：024—83680180（市场部）　83680265（社务室）
E-mail：neuph @ neupress.com
http：//www.neupress.com
印 刷 者：沈阳航空发动机研究所印刷厂
发 行 者：东北大学出版社
幅面尺寸：185mm × 260mm
印　　张：20.75
字　　数：505 千字
出版时间：2013 年 10 月第 1 版
印刷时间：2013 年 10 月第 1 次印刷
组稿编辑：郭爱民
责任编辑：李　佳　潘佳宁　　责任校对：佟　仁
责任出版：唐敏智　　封面设计：刘江旸

ISBN 978-7-5517-0455-7　　定　　价：38.00 元

# 前言

专家系统作为人工智能的一个分支，已经得到了广泛的应用。专家系统外壳或工具极大地提高了建造专家系统的效率。Pro/3 是 Jens Hintze Holm 所研制的基于 Visual Prolog 6.2 的专家系统外壳。Visual Prolog 是基于 Prolog 语言的可视化逻辑程序设计语言，是 Prolog 开发中心 PDC（Prolog Development Center）推出的基于 Windows 平台的智能化编程工具。

Pro/3 包含由事实、逻辑规则（Prolog 类型推理）等组成的知识模型。这些模型可用于进行事实集合上的统计和其他运算，也可以对带有置信度、概率和模糊集的非精确规则进行推理。Pro/3 可作为一个反向链系统来使用，此时，通过解释知识库中的事实和规则，给出所有查询及分析的结果。利用前向链结构，Pro/3 可一次性地推演句子，并将推演出的句子存储在知识库中，这对于大规模的知识库及问题求解，提高了推理性能。Pro/3 也是一个产生式系统，其中知识库作为动态存储。Pro/3 的应用涉及了非精确分类（股票选择、市场状态评估、医疗诊断及网络诊断），带有确定性规则及规则敏感应用的事件预测等。

Pro/3 作为一个产生式专家系统外壳，为研究、建造专家系统提供了一个有效、实用的平台。Pro/3 中所实现的英语自然语言句子的处理机制及非精确规则的引入，增强了专家系统非精确推理的应用功能。Pro/3 中尚有许多机制与功能有待于我们去学习、研究及应用。探索 Pro/3 对于建立辅助决策支持系统具有一定的现实意义。

鉴于此，本书系统介绍了专家系统外壳 Pro/3 及其应用，与大家共同学习、交流。希望对我国在这一领域从事教学、研究及应用开发的同行有所帮助，为提高专家系统在我国的应用水平、深化信息化应用做出一份贡献。

感谢鞍山供电公司徐兴岩高级工程师在本书部分章节内容录入中所做的工作。

由于编者水平所限，在编著过程中，难免有错误之处，真诚希望谅解并给予批评指正。

编　者

2013 年 7 月于鞍山

# 目 录

# 第 1 章　Pro/3 概述

## 1.1　Pro/3 是什么及其可以用于什么

这里首先强调的是解释 Pro/3 概念，而不是解释 Pro/3 如何做事情。Pro/3 是 Norway 的 Jens Hintze Holm 所研制的基于 Visual Prolog 6.2 的产生式系统类型的专家系统外壳，Pro/3 系统最简短可能的描述被给定在图 1.1 中。Pro/3 中最基本的概念是事实的概念，理解事实是理解其他 Pro/3 概念的先决条件，见图 1.2。

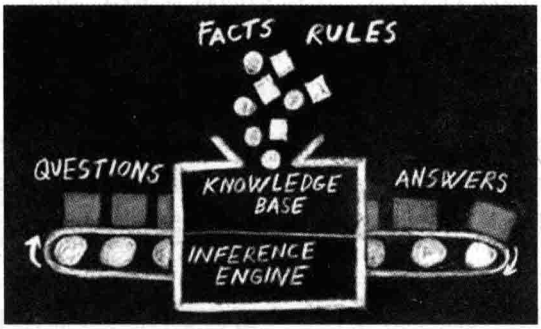


图 1.1　Pro/3 基本系统示意图

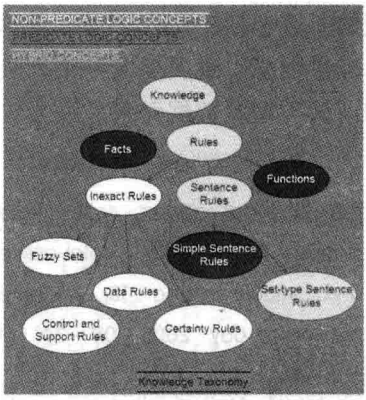


图 1.2　Pro/3 基本知识概念图

Pro/3 可以用于建造专家系统，即在问题领域可执行像专家一样推理的应用，以实现给一个非专家提供咨询意见或辅助一个专家的目的。推理的模型是基于可以包括大量事实、带有大量附加功能以处理事实集合上的统计和其他运算的“crisp”（“脆的”）逻辑规则（Prolog 类型推理）及非精确推理规则以处理确定性、可能性（确定性规则）和模糊集规则（模糊集隶属度函数），即处理非精确分类。

Pro/3 具有可以包括大量事实、“脆的”逻辑规则（Prolog 类型推理）等组成的知识模型。这些模型可用于进行事实集合，以及带有置信度、概率和模糊集的非精确推理规则上的统计和其他运算。Pro/3 可作为一个反向链系统来使用，此时，通过在运行中解释知识库中的所有事实和规则，给出所有查询的结果。然而，这种方式在某些规模较大的知识库中是不切实际的。取而代之，Pro/3 使用一种一次性地推演出所有句子并存储它们在知识库中的前向链超级结构。

Pro/3 也是一个产生式系统，其中知识库作为工作存储。一般来说，Pro/3 可能的应用

涉及了非精确分类（股票选择、市场状态评估、医疗诊断和网络诊断）、带有确定性规则和规则敏感应用的事件预测等。

下面简要介绍 Pro/3 中知识表示概念和其推理引擎原理，其例子来自于（一个已排序的）在 Oslo Stock Exchange（OSE）交易的 70 个股票的股票选择模型。

## 1.2 事 实

Pro/3 中的事实被称为句子。与句子相一致，事实可以表示为英语自然语言句子。对应 Pro/3 的事实有两种类型的句子：其一，带有一个主语和谓词的句子；其二，带有一个主语、一个宾语和一个谓词的句子。

句子结构是基于实体关系数据模型的，因此主语和宾语属于定义的实体类型，而谓词属于定义的谓词类型（一元谓词类型用于主-谓句子，而二元谓词用于主-谓-宾句子）。实体类型由数据元素类型（谓词类型可能有或没有数据元素类型）的一个固定组合组成。数据元素类型属于预定义的域的集合，这些域包括：整数、数值（实数）、串和标识符等。数值的子域包括：XML 日期、XML 时间、日期时间、持续时间、置信度、概率、增量因子、减量因子、隶属度及一个列表域补集。因此 Pro/3 有三个级别的符号结构：句子、实体和谓词类型及数据元素。

例子：

```
the financial report with ticker "ODF", operations result 26.0, net income 18.0,
current assets 325, total assets 1972, current liabilities 135, equity 702, currency US Dollar
and sequence no 10 is published for the accounting period with year 2005 and period 2 with
publication day "2005 -08 -24"!(1).
```

是一个主-谓-宾类型的句子，其中主语是 financial report 实体，宾语是 accounting period 实体。financial report 实体类型有九个数据元素类型（ticker，operations result，net income 等），accounting period 实体类型有两个数据元素，而 is published for 谓词类型有一个数据元素（publication day）。（注意句子、句子规则和函数定义的所有自然语言表示在 Pro/3 中以！为结尾）。

（1）对应的 Prolog 事实：

```
plsPubFor(eFinRep("ODF",26,18,325,1972,135,702,iUSD,10),eAccPer(2005,2),"2005 -08 -24")
```

作为术语在知识库中表示如下：

```
cmp(":; -",cmp("plsPubFor",cmp("eFinRep",str("ODF",int(26),...,int(10))),cmp("eAccPer",
int(2005),int(2)),str("2005 -08 -24")),atom("true"))
```

句子中谓词类型和实体类型的组合被称为句子类型，因此（1）属于 financial report is published for accounting period 句子类型。

Pro/3 中，自然语言格式知识和内部格式知识之间的转换由术语和句子模型而便利地实现。术语是术语学的事实集合，其阐述了如：US Dollar 是一个标识符（内部数据为

iUSD），而不是两个变量 US 和 Dollar（在 Pro/3 中以大写字母开始的字通常表示变量）。

句子模型是定义其他元素之间、谓词类型、实体类型和数据元素类型的事实集合。对于 Pro/3 声明的所有知识即规则和事实（包括术语、句子模型和其他元知识事实）存贮在称为知识库的数据库中。知识库或者由一个或多个 Visual Prolog chainDBs 实现，或者由 Pro/3 借助于 ODBC 接口访问的 SQL 数据库实现。

## 1.3 查　询

Pro/3 中有两类查询，即与一组句子合一的查询及与一个单值合一的查询。句子查询与带有变量组成句子部分的句子具有相同的结构——或者隐式地、显示地用变量（或一些变量）替代部分句子，或者给出一些条件到句子中的一个或者多个数据元素类型。

例子：

```
which financial reports with net income greater than 10.0 are published for what accounting periods with year 2004?(2)
X is published for Y for publication day greater than "2005 -02 -16"?(3)
```

（2）转换为 Prolog 查询：

```
plsPubFor( eFinRep( _, _, X1, _, _, _, _, _, _), eAccPer( X2, _), _), X1 >10.0, X2 =2004
```

其中细节可以由 Prolog 类型推理引擎解决。查询表达式将与满足给定 net income 和 year 条件的 financial reports are published for accounting period 句子（事实）合一。注意疑问词［which，what 和 who（m）］指示后面的实体类型是变量。这是基本“含糖”的语法，因为 which 和 what 可以被从（2）中删除而不改变其意义。使用显示变量（X 和 Y）的（3）有不同的解释，因为任何实体可以与 X 和 Y 合一（不仅仅是 financial reports 和 accounting periods）。

单值查询返回（合一）了一个单值（或一个单值系列，如果是非确定的）。例如，一个句子中给定数据元素的值。其他的一些变量返回了若干满足给定条件的句子或一个函数调用（见 1.5 节）返回的值。单值查询是重要的，因为非精确规则查询类型包括一个如此查询（见 1.7 节），此规则使用直接回答（或借助于一个映射）作为它的返回值。

例子：

```
what equals the value of net income for financial report with ticker "ODF" which is published for accounting period with year 2005 and period 2? (4)
what = faculty given integer 5? (5)
what equals today?(6)
```

使用上述例子中用于句子查询的相同的推理逻辑可以解决（4）的查询。（5）是一个对用户自定义函数的调用，使用通常的 Prolog 类型推理（= 和 equals 是同义词）也可以解决。（6）是对内置于推理引擎的 today 函数的调用。

## 1.4 简单蕴含规则

句子规则定义一个句子如何被从其他的句子中推演而导出。

OSE 的股票以 NOK（挪威克朗）定价，而在某些情况下，金融报告以其他的货币给出（如例子 1 有 US Dollar 计价的数字）。为了计算收益和其他指标，更容易以不同货币数字进行比较，需要转换所有数字到 NOK。可适用的转换率被一组事实给出：

exchange rate with rate = 6.52 and currency = US Dollar is quoted for trading day "2005 - 08 - 24"!

NOK 等价的金融报告被以新句子类型 nok - equivalent financial report is published for accounting period 表示。这个句子类型是由如下规则导出的：

if the financial report with ticker T, operations result OR, net income NI, current assets CA, total assets TA, current liabilities L, equity E, currency C and sequence no N is published for accounting period with year Y and period P with publication day D, the exchange rate with rate R and currency C is quoted for trading day D, then the nok - equivalent financial report with ticker T, operations result( OR * R), net income ( NI * R), current assets ( CA * R), total assets( TA * R), current liabilities( L * R), equity ( E * R) and sequence no N is published for accounting period with year Y and period P with publication day D! (7)

注意以大写字母开头的字代表变量（除非显示声明为标识符）。借助于如下 Prolog 风格的表示，此规则被转换为一个术语：

```
plsPubFor( eNfinRep( T, OR, NI, CA, TA, L, E, C, N) , eAccPer( Y, P) , D) : ; -
plsPubFor( eFinRep( T2, OR2, NI2, CA2, TA2, L2, E2, N2) , eAccPer( Y2, P2) , D2) ,
plsQuoted( eExRate( ( R3, C3) , D3) ,
T = T2, OR = ( OR2 * R3) , NI = ( NI2 * R3) , CA = ( CA2 * R3) ,
TA = ( TA2 * R3) , L = ( L2 * R3) , E = ( E2 * R3) , C = C2, N = N2,
Y2 = Y, P2 = P, D = D2, D3 = D2.
```

当写规则时，使用自然语言文本格式，正如（7）所示，通常是不实用的。此规则难读且易于出错，如由于拼写错误或其他原因而错误地省略了数据元素。对于许多用户定义变量的需求也有一些问题。一个更好的办法是使用 Pro/3 的图例规则编辑器，利用它，规则可以被画成树。规则成分如句子类型、数据元素类型和操作等可以在一个画板上采摘。对应（7）的规则树显示在图 1.3 中。

## 1.5 函 数

函数主要用于保持句子规则之外的计算逻辑性。因而使规则更简洁、带有较小的冗余度（其中相同的计算逻辑将不得不在不同的规则中重复）。这些理由在 1.6 节中将变得比较明显，其中记账日期（由年和季给定）需要关联到实际日期。图 1.4 中的简单规则为此

目的而引入，此规则使用了两个函数 accounting period start day 和 accounting period end day。

函数由一个声明（函数的可视部分，可认为是句子模型的一部分）和一个或多个定义（函数的内部部分或子句）组成。

例子：

声明：

```
the deterministic function with NL - name accounting period end day, domain name XML_DATE
and data element list year and period is in the sentence model with segment name "
OSE"!(8)
```

定义：

```
accounting period end day is defined by:;
this period = 4;
Date = xml date string given year this year, month
number 12 and day number 31;
return value = Date!(9)

accounting period end day is defined by:;
this period < > 4;
NextPeriod = (this period + 1);
NextDay = accounting period start day given year
this year and period NextPeriod;
Date = (NextDay - "P1D");
return value = Date!(10)
```

accounting period end day 函数有两个定义（子句），每个都被一个 is defined by 语句及一个返回值语句构造（由！终止——它不是一个 cut）。其他的语句由；终止。在（9）中倒数第三个语句是调用另一个函数 xml data string，此函数转换三个整数到一个日期（XML 串格式）。（10）中倒数第二个语句从一个 date 型变量中减去一个值（XML 格式 period 文字），并且赋值（绑定）此结果到变量 Date。this 是表示函数（同样的数据元素类型（名称）可能扮演着不同的角色，因此如果不限定，则产生二义性）内部形式参数的数据元素类型的限定词。

对应（9）的 Prolog 格式定义（见下面）表明函数可以由 Prolog 类型推理引擎直接解释：

```
fAccPerE( FAccPerE, FAccPerE_aYear, FAccPerE_
_aPeriod) : ; -
FAccPerE_aPeriod = 4,
p3_fDtXml( Date, FAccPerE_aYear, 12, 31),
FAccPerE = Date.
```

此术语已经赋值内部名 fAccPerE 给 accounting period end day 函数，而它的两个参数数据元素类型分别被称为 aYear 和 aPeriod。

## 1.6 集合类型句子规则

图 1.3 中的规则定义了基于发布日期现行比率的 financial 报告数字到 NOK 的转换。这个日期经常是比考虑之中的会计期间结束后多出一个月，取而代之，对于这个期间（季）使用平均比率是更有意义的。于是，推演出像 accounting period has average exchange rate 句子类型的规则是需要的。

集合规则推演由其他句子集合中导出的句子。对于眼前这个问题，一个 accounting period has average exchange rate 句子（对于每组 65 个左右属于相同的会计期间和货币的 exchange rate is quoted 句子）需要被推演。指定这些的规则显示在图 1.5 中。

集合规则使预定义的 Pro/3 过程中的一个过程与推演出的句子类型中至少一个数据元素类型关联。这表面上看像是一个函数调用，但有一个不同的解释。在图 1.5 规则中，数据元素 rate 被绑定到 AVERAGE_OF 过程。此过程有一个数值域的被绑定到来自于条件 exchange rate is quoted 句子类型的 rate 数据元素类型的形式参数。accounting period is defined 和 exchange rate is quoted 句子的与（AND）的（连接的）集合被称为条件集。

让我们看一下推理引擎如何解决查询 which accounting period has what average exchange rate? 的某些细节：

查询 pHas（eAccPer（Z1，Z2），Z3，Z4）（11）合一了图 1.5 中的结论，并且在下一步合一了它的条件。然而与像（7）的简单规则不一样，集合规则条件不能由 Prolog 类型推理引擎直接解决（尽管技术上它是知识库中的一项）：

p3_iRsta（str1，str2，str3，Var1，Var2，Var3，Var4），此项包括了三个信息片段：

- str1 是解决此条件集：pIsDefined（eAccPer（X1，X2），X3，X4），pIsQuot（eExRat（X5，X6），X7）），X7 > = X3，X7 < = X4 的查询项的一个串表示。
- str2 是对应推演出的数据元素类型：［Y1，Y2，Y3，Y4］的变量（一些项）的一个串表示。
- str3 是赋值给推演出的数据元素类型：［X1，X2，p3_rAvg（X5），X6］的表达式（一些项）的一个串表示，这里 p3_rAvg 是 AVERAGE_OF 过程的内部名。

推理引擎识别出 p3_iRsta 头部，解开并取出条件集，调用引擎的另一个实例以确定来自这个查询的实际条件集。然后，这个句子的集合及赋值的列表用于（11）中原来的查询变量 Z1 - Z4 与来自于集合（借助于 Var1 - Var4 变量）所计算的值的合一（查询匹配）。

图 1.5 中的规则是一个统计类型集合规则的例子，集合规则支持 13 个不同的过程：average，mean，sum，standard deviation，maximum，minimum，start value，end value，start point，end point，maximum point 和 minimum point 等。最后六个过程把条件集看作一个有序集（由条件中一个指定排序的数据元素类型定义的，例如 trading day）。还有一个简单返回条件集大小的 count 过程。第二种类型集合规则大致遵循了统计集合规则的模式，然而，对于导出推演的句子的过程是更复杂的，这里不予讨论。

规则的解释涉及了多种条件集（在由此规则推演出的句子被从此集合导出之前）的操

作。除了某些 selection 操作之外，这些操作包括 interpolation/extrapolation［例如在一个时间系列类型的上下文中补偿缺少的句子（值）］、与置信度和隶属度（见 1.7 节）一起经常被使用的 ranking 及在此集合中数据元素类型之间的 correlation 测定。

## 1.7　非精确规则

OSE 股票选择模型包括了 125 个句子规则，这些规则导出了大约相同数量不同的句子类型。这些句子包括股票选择指标、移动平均、波动率和收益、价格增长测量等，这些形成了与选择有吸引力股票目标相关的“脆的”知识的主体。这些知识是“脆的”，因为它们是由较好定义的数值方法导出的，本质上不带模糊的元素和主观性。

当向基于“吸引力”（attractive）的排名股票的目标进一步进行时，不久，人们将认识到缺乏一个通常可接受的、较好定义和客观的方法。有几种常见的方法和各种各样的经验规则，但是没有把它们组合到一起的数值方法。这其实相当明显，因为股票的吸引力是对未来事件可能性的测量，这些通常对于预测是非常困难的。然而，一个“好”方法仅需要产生略优于可用的随机方法的一种排名。OSE 模型使用的主体表示为模糊集经验规则的组合。

在传统“脆的”逻辑中的一个基本假设是某些事物（属于某些领域的一个元素）或者是一个给定集合的成员或者不是其中的一个成员。在后者情况下，它是一个补集（该领域中）的成员。在模糊集中情况不是如此。模糊集成员是由这个集合的隶属度函数所给定的隶属度限定的。这个隶属度函数是一个从一个元素（Pro/3 中由一个或一些参数所表示的）到范围［0，1］中的一个实数的映射。0 意味着这个元素肯定不是此集合成员，1 意味着这个元素肯定是此集合的成员，而在范围〈0，1〉中的值意味着在一定程序上（值越高，隶属度越高）此元素是一个成员。一个元素既可是一个模糊集的成员也可是它的补集的成员。

OSE 模型包括了吸引力股票（attractive stocks）的模糊集，attractive stocks 有两个参数：ticker（标识此股票）与 evaluation day（指定吸引力被评估的日期）。一个股票在 2005 年 9 月 5 日可能非常有吸引力（基于在那个日期已知的指标），而一个月后，此股票是相当没有吸引力的。此模型对于一个历史日期系列评估吸引力是非常有用的，因为在这些日期之后的不同期间，它可以用于自动关联关于持有此股票实际赢利的这些评估（参考 1.6 节关联类型的集合规则）。可测量的关联服务于模型验证和改善的目的。

模糊集和确定性规则属于 Pro/3 中被称为非精确规则的知识分类。一个非精确规则是一个返回一个简单值（典型的是一个置信度或隶属度）的函数，此函数通常有一个或多个输入参数。对此函数的一个引用（调用）的语义解释是：① 模糊集情况：此函数是模糊集隶属度函数，其返回值是由实际参数所标识的集合元素的隶属度；② 确定性规则情况：此函数和实际参数值是一个命题，其返回值是一个置信度，即这个命题的确定性的测量。

确定性规则和模糊集对应着不同的不确定性源。确定性规则一般用于问题域的知识是不精确的地方，而模糊集所用之处知识分类（知识概念）是不精确的。非精确规则大多是调用各种上下文中其他非精确规则，组合或在其他方面以不同的方式使用所调用规则返回

的值。规则调用的结构是一个直接的非循环图。图 1.6 显示了吸引力股票图的一个小子集，即定义 stocks with good result per share growth 的模糊集部分，此图的树表示显示在图 1.7 中。具有数据规则的唯一例外，所有非精确规则至少调用一个其他规则。而数据规则可以作为一种称为上下文调用的规则调用类型。一般，它们不调用其他规则，因此它们是图中的终止节点。数据规则的目的是返回一个值，此值基于或者是对知识库的查询（单值查询）或者是交互地向用户询问问题。

非精确规则以三种方式导出它们的返回值：

- 通过调用其他非精确规则并且以各种方式组合或映射这些被调用规则的返回值。模糊集操作包括：intersection 和 union（standard，algebraic product，bounded difference，drastic），complement 和 ordered weighted average。确定性规则操作包括：and，or，not，bayesian 和 combination。还有一些支持类型规则：switch-rules，parameter-rules 和 map-rules。
- 通过查询知识库（带有或不带有映射）。
- 通过向用户询问（带有或不带有映射）。

非精确规则在一组 Pro/3 格式窗口中被输入和维护。它们没有任何自然语言表示，但使用 XML 格式可以被表示为文本。图 1.8 显示了 attrative stocks 规则，通过一个 ordered weighted averaging 操作，它被定义为五个其他模糊集（stocks with good earning growth，stocks with good result per share growth）等组合的结果。attractive stocks 规则不是由其他非精确规则调用——它是由一个显示在图 1.9 中定义为 stock has buy rating 句子的句子规则调用的。

让我们分析一下推理引擎是如何解决查询的：which stocks have buy rating。在这里唯一新奇的是 buy rating 数据元素与一个调用 attractive stocks 非精确规则的合一，这可能被评估（当 ticker 和 day 参数已被来自条件句子的值合一时）。attractive stocks 规则可能仅仅在所有它的被调用规则被评估之后才被评估，递归地，直到达到非精确规则图的终止节点（数据规则）。通过处理一个单值查询或通过向用户询问（交互地）来评估数据规则。来自数据规则的返回值通过此图被向后返回，直到最终对于 attractive stocks 模糊集的一个隶属度被确定。

Pro/3 在知识库中（作为事实）存贮了非精确规则评估的完整历史。这包括了对于所有参数组合的每一个规则的评估，以及一个对应于规则图树结构的规则网络本身的评估。这些事实服务于回答某个结论为什么和如何被得到的问题之目的。

## 1.8 Pro/3：一个产生式系统

前面已经简单介绍了 Pro/3 中四种知识表示概念（知识分类）——句子类型、句子规则、函数和非精确规则。Pro/3 分析这些分类的成员之间的依赖关系、维护这些依赖作为一个被称为知识依赖图的直接非循环图。此图的方向遵循知识的 if - then 关系，它包括以下九种类型的依赖：

- 句子规则→句子类型：句子规则推演出句子类型；
- 句子类型→句子规则：句子类型是句子规则的条件；

- 句子类型→函数：句子类型在函数中被引用（函数定义）；
- 句子类型→非精确规则：句子类型在非精确数据规则的查询中被引用；
- 函数→函数：函数被函数调用；
- 函数→非精确规则：函数在非精确规则（数据规则）的查询中被引用/调用；
- 函数→句子规则：函数在句子规则中被引用/调用；
- 非精确规则→非精确规则：非精确规则被非精确规则调用；
- 非精确规则→句子规则：非精确规则在句子规则中被引用/调用。

输入到知识库中的句子类型（事实）将构成图的开始节点（不带前项的节点），而不构成任何句子规则条件部分，也不在查询规则或函数中被引用的句子类型将构成终端节点（不带后项的节点），这些是专家系统的最终结论。其他节点不可能是有意义的终端节点（句子规则必须有一个条件和一个结论，而另外的函数/非精确规则终端节点将表示不被任何其他规则或函数调用的函数/非精确规则，因此不扮演有意义的角色）。不调用任何其他函数、不引用事实及询问类型非精确规则（没有上下文调用）的函数也将构成开始节点。

例子：

OSE 模型依赖图的子集（见图 1.10）有三个开始节点（financial report is published for accounting period 句子类型，exchange rate is quoted 句子类型和 xml date sting 函数）。这个图有一个单一的终端节点：nok - equivalent financial report is published for accounting period 句子类型。图 1.10 依赖图子集中，长方形代表句子类型，圆角长方形代表句子规则，而菱形代表函数。如果图 1.10 中的知识节点表示了一个整个的知识库，不需要做进一步的努力，此知识库就可以使用，即使发布的金融报告数以千计。像 which nok - equivalent financial reports are published for which accounting periods 这样的查询可能被推理引擎快速地处理，即通过这个结论对此查询的回答开始，从那里开始向后链接，经历所有相关的规则和事实（Prolog 类型推理引擎的通常策略）。

然而，OSE 模型包括了大约 380 个知识节点及它们之间的 500 多个依赖。推演某些终端节点句子类型涉及在中间推演步骤中有超过 10 万个句子的推演。经历如此大量知识的向后链接，每次一个查询被处理，这将是非常处理敏感的（实际中是不可行的）。Pro/3 中通常的方法是更合适地将系统操作为一个前向链接的产生式系统，以使所有可能的句子被一次性地推演，然后正如图 1.11 中所显示的那样存贮在知识库（产生式系统的工作内存）中。

前向链句子推演策略是基于知识依赖图的，此知识依赖图定义了句子推演所要求的句子，也跟踪了推演的状态。这里两个附加的机制被采用，第一个：区分用户输入的句子和系统推演的句子；第二个：区分句子规则的两种状态（活动的和非活动的）。非活动的规则被推理引擎忽略，当所有可以从此规则推演出的句子已经被推演且存贮在知识库中时，才是我们想要的东西。另一方面，活动的规则总是被推理引擎考虑，在图 1.11 中第三步期间或仅当使用反向链接操作知识库时（即没有正向链接超级结构），这才是需要的。在这种情况下，这些规则在所有的时间内都是活动的，被推演出的句子不存贮在知识库中。

与输入的句子不同，在句子推演过程中（图 1.11 的步骤 1 和步骤 4），推演出的句子被系统删除和重新推演。重新推演将仅当需要时发生，即当决定一个推演句子的一些规则或句子已经变化时。例如，若图 1.10 中的 exchange rate is quoted 句子已经改变（或图 1.5