维普资讯 http://www.cqvip.com

软件工程 时装管理系统 概念模式框架 (12)

计算机科学 1995 Vol. 22 №. 1

55-60

对象管理系统中的概念模式框架*)

郭 江 廖越虹

(北京航空航天大学软件工程研究所 北京 100083)

TP311.5

摘 要 The paper discusses an important concept in the field of data bases, that is conceptual schema framework. First, the paper defines the concept of framework, and discusses two key terms : an universe of discourse (UOD) and a conceptual schema (CS). Then, we classify the instances and objects, define a paradigm describing the UOD and CS, and arrive at the conceptual schema framework (CSF). Finally, the paper discusses some problems of modeling paradigm and implementation paradigm. And the paper is closed out by discussing some applications of this framework.

关键词 DBMS, conceptual schema (CS), universe of discourse (UOD) conceptual schema framework (CSF).

一、引言

目前，一般使用三模式(外模式、概念模式、内模式)体系结构作为 DBMS 的系统框架。随着信息管理系统的发展，大多数据库的外模式和内模式都在不断地变化着。在外模式上，用户的需求和复杂性与以前相比已经有了很大的差异。对内模式而言，由于面向对象数据库(OODB)逐渐进入角色，也随之在变。因此，概念模式就变得越来越重要了，是信息系统稳定的根源[1]。

对象管理系统(OMS)是计算机辅助软件工程(CASE)环境的核心，在 CASE 环境中不仅要存储有关 CASE 环境以及软件开发过程的信息，而且是沟通各个工具的媒介，本文主要讨论 OMS 的概念模式框架。

二、框架

框架是以一种结构化的方式来安排相关概念集的一种手段[2]。描述框架的最好方法是用图来表示，该图可以是二维的，也可以是多维的。本文中，特定义一个二维框架来说明信息系统实现的某些重要方面。如图 1 所示，纵向表示：①论域(UOD)是在一个现实世界或假设世界中出现的、人们感兴趣的、要描述的所有对象(实体)的集合；②概念模式(CS)是 UOD 事件可能状态的描述，包括 UOD 的分类、规则、定律等。横向表示实现信息系统所需 UOD 成份的开发集。使用这种二维表，就可以开发概念模式框架(CSF)了[3]。下面首先分五个阶段来开发 CSF 纵向描述的 UOD 范例。

三、构建块

首先考虑术语对象及其包含的概念并考虑 UOD 的定义。一个 UOD 与真实或假设世界有关，但目前只关心真实世界。真实世界对象是每天处理的各种"事物"，如物理的程序、目标代码、文档；还有另一些概念，如天、月、小时、工作安排等。许多对象有相似的特征，但每个有它自己的标识和特点，这些相似的对象属于一个类型或类。因此，它们可以由一个组或类名来调用，而其相似特征则与类的表示联系在一起。

这些分类目前用两类术语集来描述：一是用类(或类型)来标识一组相似的事物，而用对象来标识一个类中的各种事物；另一是用对象来标识分组，而用实例标识属于该组的不同事务。这里，将选用后一类术语，因而实例意味着各种单个事物，而对象表示有相似特征的实例集。

在面向对象的上下文中，对象有特定的含义，如封装、继承、方法、消息等；而在本文中，对象的概念并不意味着是这样一个面向对象的基础。在开发 UOD 范例的最后阶段要确定它的两个成份：实例和对象，如图 2 所示。

*)国家"八五"项目资助。 郭 江 博士生，现为系统分析员，主要研究领域：软件工程、数据库、软件工程环境。

维普资讯 http://www.cqvip.com

四、UOD 中的实例

所有实例的集合构成了 UOD 的核心，意味着一个可标识的实例标识符与 UOD 中的每个实例相关，该特征在 OMS 的实现中是强加的，UOD 任何部分的描述总可归化为一个偏序表或 UOD 中所有实例集合的子集。

下面考虑怎样用实例来描述 UOD 的表示。许多情况涉及到了单个实例的使用，但是多重实例的某种组合作为一个单位是经常出现的，因而可将实例分成两类：简单的和复杂的。简单实例是不能分解成其它实例的成份集，而且在所考虑的 UOD 应用的上下文中也不能通过在其它实例上执行操作来导出或生成。复杂实例分成两类：合成的和导出的。合成实例表示其它实例的简单组合或连接，并且由这些实例标识符的组合或连接来标识。这些实例的典型说明是那些代表数据值（年、月、日值的连接）、地址值（驱动器号、路径名、文件名的连接）的实例。同样，导出实例是通过在其它实例值上执行预定操作而导出的值来标识的。一些例子如汇总（如，数量的月汇总是由日实例的值累加得到的）；统计值（平均值等）；以及计算值（如通过将单价乘以数量而得到的价格总值）。

合成和导出实例表示唯一的事物，但是这两者均不能真正独立定义，因为均依赖于形成它的简单或复杂实例。这些概念类似于数学中的独立和依赖变量，可以将简单实例看作是独立变量，复杂实例看作是依赖变量。图 3 描述了这种分类和结构。

UOD 范例开发的第 2 阶段是将实例的这种分类结合起来，并由完整的实例框架（包括简单、合成和导出部分）来描述，如图 7 所示。

五、UOD 中的对象

可以将具有相似特征的实例归为一个对象，因而对象表示了一个或多个实例。这样就可约定：每个实例必须至少属于一个对象。如果存在一个唯一的实例，即其特征与所有其它实例都不相同，那么就可由其自己构成一个单独的对象。一个实例可以属于多个对象，这依赖于对象是如何构想的。

对象是具有相似特征的实例集合的表示，更实际地看，这种特征与对象扮演的角色相关，而且描述了事件的状态。实际上，这种角色的扮演不是确定和定义对象的一个主要标准。

因此，对象具有的特征以及该对象所扮演的角色就是决定一个实例是否属于一个对象的标准。如果一个实例具有的特征和扮演的角色与多个对象相关，那么该实例就可以属于所有这些对象。图 4 描述了 UOD 中对象和实例的关系。

六、对象分类

下面描述 UOD 中对象的分类方案。

(1)语法分类。指依赖于公式表示的方式，即实例的语法。这种方法可以将对象分为简单的和复杂的。简单对象就是那些实例均为简单实例的对象，而复杂对象则是至少有一个实例为复杂实例的对象。

正如实例一样，复杂对象又可进一步分为合成和导出对象，但是使用这种对象组成的方法来进行细的分类时，必须考虑几个问题。如，一个复杂对象可以由简单和复杂实例组成吗？一个复杂对象可以由合成和导出实例组成吗？这些问题并不直接影响本文的讨论，但在使用这些概念实现系统时是非常重要的，因而有待进一步研究。

(2)语义分类。指涉及到怎样使用 UOD 中的对象，即使用的方法。下面再次考虑 UOD 和 CS 的定义。

UOD 是所有实例的集合，而 CS 则描述了 UOD 中事件的可能状态，但是最初这些定义并没有真正表达人们要表达和沟通的思想。建立通讯的最好媒介是用自然语言中的命题和断言描述 UOD 中的事件状态。在最低层次，这些断言应用 UOD 中的实例来作出，即实例级断言，然后就可以依赖于它们所涉及的实例对这样的断言进行分类和抽象。

如果若干断言描述事件的同一个状态，那么就可以作用于属于同一个对象的多个实例上。这样就可以用一个对象级的断言表示所有这样的断言，描述实例所属对象事件状态的断言。实际上，对象级的断言总可用于描述 CS，其准确性是通过构造相应的实例级断言并验证其正确性来保证的。

下面是一些用于描述 UOD 中事件状态的典型断言，①给一段代码命名。②程序有三种表示：源代码、抽象语法树(AST)、符号表。③每个程序对应一棵抽象语法树。注意，系统的目标、规则、策略和操作包含在有关系统的这些断言的一个完整集中。

自然，这样一个完整集除了简单的断言形式外还将包含断言的许多复杂形式，对这些断言的研究将导致对事件状态的理解，即 UOD 和 CS。这种语言和谓词演算的原则提供了研究和分析自然语言的理论基础，因此在表达成自然语言中的一个断言集时，

维普资讯 http://www.cqvip.com

可用于研究 UOD 的 CS，从这些原则借鉴合适的概念，就可以将这样的语言断言表达成 UOD 中对象间的语义相关集以及这些相关关系上的约束集。

在确定这些断言中的对象时并没有太多的困难，例如，源代码和符号表对象，称这些为真实世界对象。也可以进一步抽象对象概念，并约定描述这些对象间语义关联的关系也属于一个一般的类或对象，称之为关联对象，该对象的实例具有的公共特性是它们描述了真实世界对象间的关联方式。

从 CS 的观点看，关联对象和 UOD 中的真实世界对象在语义上是有差别的，前者描述了对象间语义的关联。

这里，除了行为规则（即约束）是规定在基本断言以上之外，这些规则还由特定的修饰来确定，如“必须”，“每个”，“有且仅有一个”，并给出一个称为约束的对象来描述，这些实例的共同特性是它们对关联对象实例所包含的关联方式施加了某些约束。

因此，三个语义类是：真实世界对象、断言对象以及约束对象。图 5 给出了整个对象空间的一个图，其中语法类为一个轴，语义类为另一个轴。

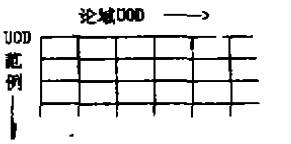

图1 CSF轮廓

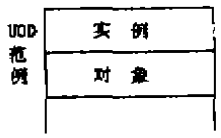

图2 UOD范例——第一阶段

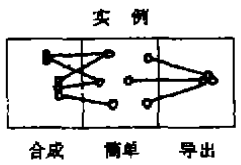

图3 实例分类

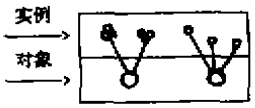

图4 对象实例关系

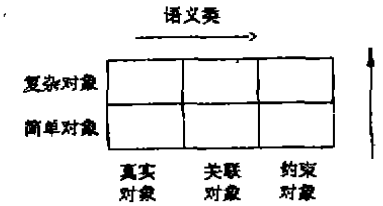

图5 进行了语义和语法分类的对象空间

UOD 范例开发的第 3 个阶段将对象的语义分类结合在了一起，该过程如图 7 所示。

应当注意，如果所有关联和约束实例都属于一个对象，那么就只有一个关联和一个约束对象。但是，也可以对这些实例进行进一步地分类，可以定义多个关联和约束对象，因此对象语义分类的一般方案是：多个真实世界对象和一个或多个关联及约束对象。

七、概念模式

UOD 范例开发的第 4 个阶段就是扩充范例，包括由真实世界对象类和其它两个对象类中的对象和实例组合而成的 CS，如图 7 所示，这个阶段是在对象分类之后进行的。

对开发范例进行总结，就可以开始确定用户 UOD 中的三类实例。这些实例在对象级归类于真实世界对象。通过对描述 UOD 事件状态的对象级断言的研究，就可以将对象分为两个语义类：关联的和约束的。CS 是通过组合真实世界对象以及关联和约束对象的实例而构成的，这些组合表达了系统的策略、规则等等。

在对象的语义进行分类之后，就可以将之用于定义 CS，也可把它表示为语义网。语义网定义为结点和弧的集合。其中，结点（圆圈）表示真实世界对象，弧（典型符号）表示关联和约束对象的实例，如图 6 所示[1]，该图不包含任何特定的模型化范例。

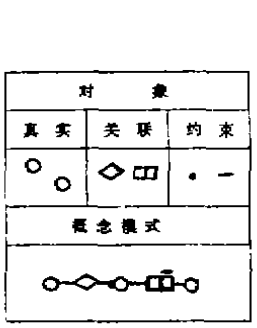

图6 一个CS的语义网表示

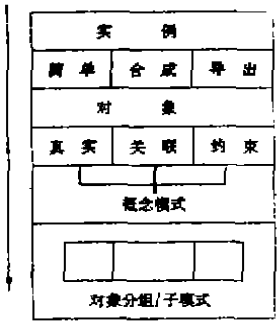

图7 UOD范例——阶段5

八、对象分组

在 UOD 范例开发的最后阶段是要确定另一个概念对象分组，这对描述 UOD 的分类并非必需，但从实际的观点看为完善该范例所必要。这个概念类似于定义合成对象的概念，确定了形成对象组的一个对象集合，对完成 CSF 的映射必不可少。如果将这种分组建立在某些语义相关的对象基础之上，就可以将它表示成 CS 语义网的邻接部分或子模式，这个过程如图 7 所示，用这个范例作为基础，可以描述 UOD 和 CS 的分类：

维普资讯 http://www.cqvip.com

·一个 UOD 可以用实例和对象来描述。

·在第一层，UOD 由简单实例组成，其中每个都是不可分但可标识的。每个实例均可指定一个唯一的 UOD 范围内的实例标识符。

·复杂实例是简单实例的合成或导出，依赖于它们形成的方式，可分为合成的或导出的。

·对象是具有相似特征的实例组的表示。每个实例至少属于一个对象，某些实例也可属于多个对象。

·对象可以进行语法和语义分类。语法分类可产生简单和复杂对象类。语义分类产生真实世界、关联和约束对象类。这些分类模式相互独立。

·CS 可以表示为涉及 UOD 对象的自然语言断言集。这些断言可以表示为真实世界类对象与关联和约束对象实例的组合。

·对象的语义分类可以将 CS 表示成语义网。

·可以识别 UOD 中对象的分组。如果一个组内的每个对象必须是语义相关联的，那么这样的分组或子模式就成了 CS 语义网表示的邻接部分。

现在，已开发了一个范例来描述论域(UOD)及其基于自然语言表示的概念模式(CS)。定义了 UOD 范例的构建块：实例、对象、断言和对象组。断言表示了对象间的语义关联，而约束则是对象上的限制。这个 UOD 范例构成了 CS 框架(CSF)的纵向分类。

下面将讨论 UOD 和 CS 的表示方式。实际上，图 6 就是 CS 的图形表示，通常用于信息模型之中。就此而言，信息模型只是一个选定的 UOD 的 CS 表示。图形表示有助于对 CS 进行系统的文档化，而且给用户交流有关 CS 的信息提供了手段。可以将 CS 以机器可读的形式表示出来，并提供工具设计成标准化的记录结构，也可以设计其它范例以满足不同的需求。

在引入一个新的范例时，将引入一些崭新的概念，因此必须对这些新引入的概念以及行为规则做新的断言。换句话说，要引入新的 UOD 及合适的映射，以表示用户的 UOD。

目前，已开发了一个范例来描述 UOD，就可以使用同一个基本范例来描述新引入的 UOD 了，然后可以讨论用户 UOD 成份之间的映射和新引入的 UOD。下面将讨论如何把这两个 UOD(模型化范例和实现范例)结合到 CSF 中。

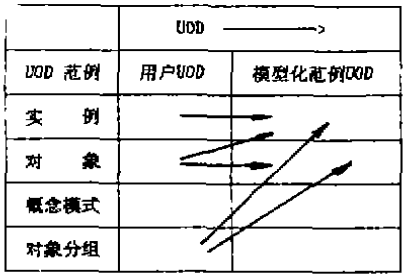

图8 将模型化范例结合到CSF中

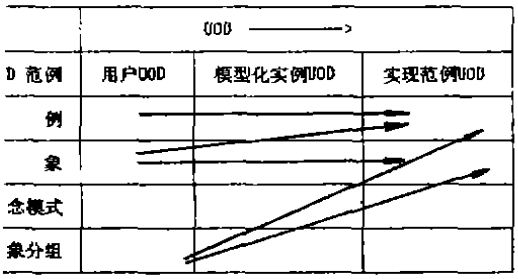

图9 将实现范例结合到CSF中

九、模型化范例

模型化范例是 CSF 的第一个横向扩充。在图 8 中，将 UOD 范例应用到用户 UOD，然后应用到模型化范例的 UOD。这个方法使得我们可以确定实例、对象、各自分类、以及模型化范例的 CS 和对象分组；用户 UOD 的 CS 由用户 UOD 的对象和实例映射到模型化范例 UOD 中相应部分来定义。

要注意下面几个问题。首先，由于模型化范例的本质特征所决定，一些 UOD 范例中的成份可能在模型化范例的 UOD 中是不可确定的。例如，如果模型化范例仅用于模型化用户环境，UOD 范例对象组上的 CS 可能没什么用处。但是，用该模型作为实现模型化范例的基础时就很重要了，模型化范例 UOD 的 CS 成了通常称为元模型的概念。其次，图 8 中的映射箭头显示了最通常的映射类型。发出箭头的格子确定了映射的是用户 UOD 的哪个成份，而箭头指向的格子表示了映射成份。箭头表明：

·用户 UOD 中的实例映射到了模型化范例 UOD 中的实例。

·用户 UOD 中的对象可以映射到模型化范例 UOD 中的实例和对象。

·用户 UOD 中的对象分组可以映射到模型化范例 UOD 中的实例和对象。

维普资讯 http://www.cqvip.com

这些情况起初可能容易混淆，但均可能发生。试考虑下面的例子：

使用实体关系(ER)模型作为范例，用户 UOD 中的真实世界对象映射到了模型化范例 UOD 中的对象实体或属性的实例上。用户 UOD 中的关联对象映射到了模型范例 UOD 中的相应对象关系上，这样就将这两个对象的实例映射在了一起。这些例子表明可能存在的映射种类，但这并没有穷尽用户 UOD 和特定模型化范例之间的所有映射。这种方法的实现者必须建立完整的映射。

十、实现范例

正如早先指出的，引入实现范例就好似引入一个新的 UOD。因此，第一步是为该新 UOD 定义 UOD 范例的合适内容。对关系范例而言，表、列、索引等等均变成了这个新 UOD 的真实世界对象。新 UOD 的 CS(有时称为关系范例中的元模型)反映了下面三个断言：表有列；列有格式；一些列可用于索引之中。一旦建立了这种新 UOD 和 UOD 范例，下一步就是将用户 UOD 的成份映射到在新 UOD 中的相应部分上。这个过程的结果如图 9 中所示，而且箭头表示了所有可能的映射。在范例中，如果没有相应部分的存在，可能缺省某个映射。

这样映射的结果是相同项的多种表示，因此必须使用唯一的实例标识符，做到一个无二义性的、一致的实现。每个简单实例至少指定一个系统范围内唯一的标识符；然后，给一个实例指定标识符或同义词。但是，主标识符必须是全局唯一的。如果给实例指定了其它标识符，那么它们必须在局部使用的上下文中是唯一的。合成实例可以由组成它的简单实例标识符来标识，实现人员也可以指定一个系统范围内唯一的标识符。根据复杂实例的形成方式，可以在存贮时指定一个系统生成的唯一标识符，也可以是导出的并且动态地用于特定的目的。这种方法保证了只要系统内部操作使用主标识符并维护一个同义词的分类，那么实例就是无二义的标识。

十一、应用

在将一个新 UOD 结合到框架中时，就在用户 UOD 和新 UOD 的成份之间建立了映射。在某个成份没有对应的部分时，这些成份仍旧没有表示出来，使得在这个范围内表示不够充分。如果将这样的范例结合到了 CSF 之中，并实现了映射，那么用户 UOD 中的许多成份将仍旧是不可映射的。例如，在关系范例中就不存在用户 UOD 中"断言"对象的直接对应部分，ER 范例的早期版本就不能表示用户 UOD 中"约束"对象的所有实例。

下面考虑可使用 CSF 作为基础来开发的四个应用。

1. 仓库。CSF 为一个仓库提供了完整的信息核，它以一种唯一可标识的方式文档化了用户 UOD 的成份及其在仓库实现不同范例中的表示。自然，必须增加额外的处理机制以提供必要的仓库服务。

2. 集成。可以考虑一个集成应用系统，既可以用相同或不同模型化范例中的两个或多个模型实现，也可以用相同或不同实现范例来实现两个现存系统。不论在哪种情况下，步骤都是相似的：

步 1：用 UOD 范例描述模型化或实现范例。

步 2：建立用户 UOD 的属性成份和在第一步建立的模型化或实现范例的 UOD 成份之间的映映。

步 3：导出和文档化由两个模型(或系统)中某一个所表示的用户 UOD 成份，并建立映射。

步 4：建立在第 3 步导出的用户 UOD 成份在另一个模型(或系统)中表示的那些成份之间的映射，确定已经在第 3 步导出的成份间的重叠部分并给用户 UOD 添加新发现的成份。

扩充的用户 UOD 现在表示了两个模型(或系统)的集成，确定了公共而唯一的区域。这种描述的简单性并不意味着集成过程很容易完成，但是在完成集成的过程中需要这些步骤。

3. 范例统一。可以将这种应用看作是一种集成，是两个范例的集成应用系统，因此这种方法可以使用步骤 1 和步骤 2 中的操作来完成。如果用这些范例实现的系统(或模型)不可用，那么步骤 2 可以由独立比较和映射两个 UOD 中的成份来代替。但是，如果它们是可用的，这种方法就能很快达到统一，确定这些范例的公共而唯一的部分。

4. UOD 翻译。一旦范例统一了，就能决定使用一个范例创建的模型翻译成另一个范例的程度，并通过直接将一个 UOD 表示的成份映射到另一个的相应部分来进行翻译。

十二、将来的扩充

CSF 确实是一个解决信息管理问题的尝试，概念可能很简单，但过程却很困难，本文所讨论的 CSF 方法是处理信息管理问题的一个具体、一致和可扩充的基础。尽管这个方法的基础具有某种程度的直觉和经验因素，但它很容易映射到更形式化的方法

维普资讯 http://www.cqvip.com

上，如一种使用谓词演算的逻辑方法。CSF 应用的一个实际例子是 Saros 公司的 Mezzanine/FileShare 产品中的集成仓库[2]。该产品建立在用面向对象范例设计的用户接口之上，内部实现使用了关系范例，有一个内部仓库，而相关的处理代码使用了一些基本的 CSF 概念。这种创新达到了两个范例间的完全对射变换以及相当好的自动映射和随后的代码生成（以 C 头文件和存贮 DBMS 过程的形式）以处理关系操作。这种方法简化了连续变化的处理及系统进化中的改进。

前面开发的 UOD 范例是 CSF 的核心，范例越完善，CSF 的潜力就越强。下面考虑范例的完整程度。

首先，在模型化原则中，当前问题是基于数据模型化和过程模型化的两个方法之间存在差别。一般而言，CS 方面是从数据模型化的角度说明这个环境，因此处理模型化方法的支持者说 CSF 对数据模型化很好，但对处理模型化怎样？我们怎样使用这些概念来对待处理模型呢？回答是提供一个新的办法，其中处理这一概念抽象为一个对象，而各单个处理就是一个实例，有点儿类似确定关联和约束对象。实际上，可以通过引入一个新的与处理有关的 UOD 来进一步抽象概念并将之结合到 CSF 中，UOD 可以建立在与处理模型化相关的"输入-处理-输出"范例基础之上。在这种情况下，输入和输出映射到用户 UOD 中的对象分组上，而每个处理变成了新对象"处理"的一个实例，它对这个新 UOD 是唯一的。前面已经使用了这个概念，只是在考虑导出实例的形成时没有给以它明确的定义。一个导出实例是一个处理的输出，该处理过程在输入到该处理的实例集上进行操作。这个方面如何模型化并将之结合到 CSF 中将使实现者能把所有面向计算的处理连接到 CSF 中是值得进一步研究的。

上面讨论的断言（特别是对象语义分类方面的断言）例子显示：总的断言集占完整 CS 的一个相当大的部分。但是，不可能建立一个包罗各种情况的全集。如果能建立的话，这将形成信息系统的一个完整基础。但一些断言包含了对象和相关量词（类似于语法中的形容词和副词）；另一些包含了条件子句、基于布尔运算符的断言的表达式等等。

为了使 CSF 基础完整，必须分析所有这些断言，引入适当的对象类，扩充到 UOD 范例中。但是，从一个实际的角度看，如果一个部分断言类型的范例覆盖了断言全集的一个百分比，而该断言全集对开发的目的是足够的，那么建立在这个基础上的实现方法将是可以接受的，不能因为有一小部分断言不能进行适当的模型化和实现就不做下去。使用这个标准，甚至仅是一个简单的断言集（该断言集可以使用语义对象分类，真实世界、语义及约束对象来覆盖）就能比传统方法好得多。

CSF 给新信息系统的实现提供一个飞跃，同时对 CSF 的扩充（横向是增加 UOD 的新类型，纵向是细化 UOD 范例）可以相对直接地对系统进行扩充，而不需要任何大的重新实现。因此，这就成了进化、开放系统的核心，并以一种增量方式说明新的问题。

参考文献

[1] Gadre, S., A Reference Framework for Conceptual Schemas, Presented at ISO/IEC Meeting on Conceptual Schema Facilities/Common Data Modeling Facilities, March 1992

[2] Gadre, S., A Framework for Conceptual Schemas, Database Programming and Design, July 1993

[3] Concepts and Terminology for Conceptual Schema and Information Base, ISO/TC97/SC5/WG3, March 1982

[4] Principles of Semantic Networks, Sowa, J. (ed), Morgan Kaufmann, 1991