面向服务业的通用业务知识模型与逻辑表示

周 平[1,3] 王卫民[1,3] 罗伟民[2] 陈其铭[2] 郑宇飞[3] 曹存根[3]
(江苏科技大学计算机学院 镇江 212003)[1] (中国移动通信集团广东有限公司 广州 510100)[2]
(中国科学院计算技术研究所 北京 100190)[3]

摘 要 现有的知识管理方法多种多样,但是不存在统一的解决方法。在服务业中,借鉴都柏林核心的思想,建立一种普适的结构模型,采用分层的思想,对服务业知识进行建模。为了更清楚地表述模型,引入动态描述逻辑(DDL)的方法描述模型,对模型中的动作及模型的状态进行表示,使知识更利于计算机严格地操作。

关键词 通用业务模型,主题,动态描述逻辑,动作
中图法分类号 TP302.1 **文献标识码** A

Dynamic Description Logic Based Approach to General Service Knowledge Modeling

ZHOU Ping[1,3] WANG Wei-min[1,3] LUO Wei-min[2] CHEN Qi-ming[2] ZHENG Yu-fei[3] CAO Cun-gen[3]
(College of Computer Science, Jiangsu University of Science and Technology, Zhenjiang 212003, China)[1]
(Guangdong Company, China Mobile Limited, Guangzhou 510100, China)[2]
(Institute of Computing Technology, Chinese Academic of Sciences, Beijing 100190, China)[3]

Abstract There is no uniform solution to knowledge management though various kinds of methods have been proposed. This paper introduced a sort of general structural modeling method of service industry drawing on the experience of Dublin Core Metadata Elements with a thought of hierarchy, for modeling knowledge of service industry. In order to describe the model clearly, Dynamic Description Logic(DDL) was used to model actions and state changes in the model, which can make knowledge description and management more rigorously.

Keywords General service modeling, Subject, Dynamic description logic, Action

1 引言

资源是一切可被人类开发利用的客观存在,是任何可以标识的东西。

为了从分散的、多种多样的资源中提取知识,需要对资源进行数据化描述,将各种资源结构化,使分类准确、有效,便于用户对知识的管理和使用。描述资源的过程中,采用元数据的思想。

元数据是一种结构化的、用以描述数据的数据,是一组独立的关于资源的说明[1,2]。元数据由于可以解决资源的语义描述问题,易于形式化,提高了计算机可操作的可能性,因此采用元数据的方式易于实现资源的结构化以及资源的计算可理解,最终实现资源的有效管理和知识的散播、重用。

都柏林核心元数据就是一组基础元数据,包括15个元素[3],可以通过组合、扩展,用于描述不同领域的资源。

与都柏林核心类似,我们提出适用于服务业的一组元数据,称作"主题",它用以描述经过考察的金融、IT、医疗、气象等行业的服务产品信息的相关知识资源。同时在主题概念下提出"摘要",它是描述一个业务主题的一个方面,用以连接知识库与问题空间。

描述逻辑是一种知识表示的形式化语言,用来表示和推理某一领域的知识,目前被广泛应用于知识表示及计算机科学的许多领域中[4-6]。

使用描述逻辑可以建立某一领域的概念模型。概念模型中包括3种对象:类、个体和关系。在描述逻辑中,用概念表示类、用个体名称表示个体、用角色表示关系。通过概念和角色,我们可以描述领域中个体的性质:个体与概念之间的实例关系,个体与另外一个个体之间的关系,以及概念之间的等价、包含与不交的关系。

描述逻辑[6,7]表示的是领域在某一时候的状态,表示的仅是静态信息,因此只能用来表示静态知识库。为了表示随动作变化的动态信息,Frank Wolter等提出动态描述逻辑[8]。在动态描述逻辑系统中,动作是一个模态算子,将动作与状态的变化联系起来,动作可引起状态的变化。例如,在通用业务模型中,假设动作"小李开通动感地带业务",由当前状态 w_1 通过该动作到达状态 w_2,"小李开通动感地带业务"用模态词 α_1 表示,w_1,w_2 通过 α_1 是可达的,w_1,w_2 的信息用描述逻辑表示。

到稿日期:2010-08-18 返修日期:2010-12-09 本文受国家自然科学基金面上项目(60773059)资助。

周 平(1985—),男,硕士生,主要研究方向为知识获取,E-mail:zhoupingaust2008@yahoo.cn;王卫民(1977—),男,博士,主要研究方向为人工智能;罗伟民(1975—),男,博士,主要研究方向为移动通信网络、人工智能;陈其铭(1978—),男,博士,主要研究方向为移动通信网络、人工智能;郑宇飞(1978—),男,硕士,助理研究员,主要研究方向为知识管理、Web服务;曹存根(1964—),男,研究员,博士生导师,主要研究方向为人工智能。

在通用业务模型中，利用动态描述逻辑描述建立的元数据模型，可清晰地表示模型中的动作，解释模型所处的不同状态，使知识最大限度地便于计算机处理。

本文第2节介绍一种服务业通用业务模型；第3节利用动态描述逻辑对给出的业务模型进行描述；第4节给出在模型上基于动态描述逻辑的推理举例说明；最后给出本文的小结。

2 通用业务模型

各种各样的服务企业均具有以下两个特征：

1)业务池中含有大量的业务；

2)客户可以对每个业务施加若干动作，如“开通”、“取消”、“设置”、“查询”等。

服务企业，可提供给客户各种业务，例如中国移动提供“全球通”、“动感地带”、“神州行”业务。各业务下分种类繁多的子业务，业务与业务之间的关系错综复杂。每种业务在不同时刻处于不同状态，不同状态空间之间的变化与联系使得服务业务形成一个动态变化的领域。领域的动态性体现在一些关键的动作上，比如业务领域中，开通业务和取消业务的动作会使整个业务池的状态发生改变。

能否形成一个统一的、严格的业务动态描述方法和模型，是本文的研究重点。

2.1 模型概述

在利用现有的行业知识时，存在两个问题：1)客户或者知识获取者无法准确获取相关知识；2)知识提供者无法提供准确、全面的知识。为了解决这两个问题，需要对行业知识进行有效的管理。

我们提出“服务产品”这样一个概念，将其定义为“无形的、给资源利用者带来某种利益或满足感的有偿或无偿的一种或一系列资源。”

我们将服务进行分层，对各种子服务进行描述，在服务产品下提出“业务”的概念，“业务”由各种主题元数据描述。通过分层，可以使建立的模型结构清晰、层次分明，利于知识的结构化，便于最大程度地实现知识管理。

本模型中，“主题元数据”构成了整个模型的框架，我们可以通过“主题”来结构化地描述业务。但是需要详细说明具体的业务内容描述或操作，以便于服务商利用、维护信息资源，也便于客户更好地了解信息资源。

在业务的具体实施过程中，用户会向服务商提出许多问题，经记录形成“问题空间”。“问题空间”有助于服务商改进服务资源，提升服务等级。

为了解决“问题空间”中的实际问题，需要在已有的知识库当中找出合适的答案，快速有效地解决问题，所以在“问题空间”与知识库之间就存在一种交互。为了解决这种交互，在“主题”一级下，我们提出“摘要”的概念，用来充当知识库与问题空间之间的桥梁。其目的是为了准确、有效地描述问题，且快速、准确地解决问题。

“摘要”的一般形式为

〈业务〉主题(标签)

在“摘要”的一般形式中，“〈业务〉”表示所要描述的业务对象，比如“彩铃”、“GPRS”这样的具体业务。“(标签)”用来对“摘要”进行分类，描述同属于一个摘要的不同情况，比如彩铃资费介绍(海外)、彩铃资费介绍(港澳)。

2.2 术语和定义

(1)主题(subject)

对服务资源抽象得到的一组元数据，用以准确、简练地描述服务产品资源。

(2)主题元数据(subject metadata)

用以描述信息资源各种形式特征的数据集的数据。

(3)一级主题元数据(first subject metadata entity)

唯一标识一个数据集所需要的基本的最少数量的元数据元素。

(4)二级主题元数据(second subject metadata element)

建立完整的数据集所需要的全部元数据元素。

2.3 模型结构

下面给出建立的业务模型的简要层次结构：

```
〈? xml version="1.0" encoding="UTF-8"?〉
〈xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="serviceModel"
xmlns="serviceModel"
elementFormDefault="unqualified"
attributeFormDefault="qualified"〉
〈xs:element name="业务"〉
  〈xs:complexType〉
    〈xs:sequence〉
      〈xs:element name="名称"〉
        〈xs:complexType〉
          〈xs:sequence〉
            〈xs:element ref="正式名称"/〉
            〈xs:element ref="交替名称"/〉
          〈/xs:sequence〉
        〈/xs:complexType〉
      〈/xs:element〉
      〈xs:element name="介绍"〉
        〈xs:complexType〉
          〈xs:sequence〉
            〈xs:element ref="资费介绍"/〉
          〈/xs:sequence〉
        〈/xs:complexType〉
      〈/xs:element〉
      〈xs:element name="开通"〉
        〈xs:complexType〉
          〈xs:sequence〉
            〈xs:element ref="开通方法"/〉
          〈/xs:sequence〉
        〈/xs:complexType〉
      〈/xs:element〉
      〈xs:element name="使用"〉
        〈xs:complexType〉
          〈xs:sequence〉
            〈xs:element name="取消"〉
              〈xs:complexType〉
                〈xs:sequence〉
                  〈xs:element ref="取消方法"/〉
                〈/xs:sequence〉
              〈/xs:complexType〉
            〈/xs:element〉
```

```
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

从给出的简要模型中可以看出，我们对元数据的描述采用了分层的结构。这种结构对于资源的细化有非常好的效果，可以更清晰地描述资源，达到计算机可处理资源的目的。

在业务模型中，客户可以开通、使用各种业务，可以执行多种不同的动作。在这样一个变化的领域中，动作引发状态之间的变化，动作的发生使得业务模型领域具有动态性。所以我们利用动态描述逻辑来描述模型，反映模型的真实变化状况。

3 通用业务模型的动态描述逻辑表示方法

动态描述逻辑将动作作为模态词来表示动作信息。这种方法可以直观地表示动作，但是不能够描述动作的性质。每个动作只能用一个单一的符号描述，系统无法获得动作的信息，并且不能够描述显示中动作与动作之间的关联。所以，结合描述逻辑与动态描述逻辑，引入文献[3]中的3个层次的思想来描述通用业务模型。

3个层次为动作层、中间层、对象层。考虑“小李开通动感地带”这个动作，在动作层中采用具体化的方法描述动作，抽象开通动作类 Deploy，这一层描述业务模型中所有开通动作的静态信息；对象层中用描述逻辑描述各个对象，并抽象成用户类 Customer、业务类 Service，这一层描述的是业务模型某一时刻的状态的静态信息；中间层将这两层联系起来，包含动作层中的开通动作个体形成的模态词。模态词作用在对象层，形成动态概念与断言，用来描述业务模型的动态信息。其图示如图1所示。

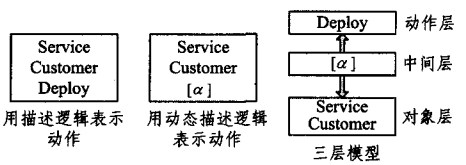

图1 模型的动态层次结构

3.1 动作的表示

在业务模型中，我们考虑开通与取消两个动作。假设服务商提供4种业务，分别用 $sv1, sv2, sv3, sv4$ 表示；有3个客户，分别用 $cs1, cs2, cs3$ 表示；假设每种业务只可以提供给一个客户，且每个客户只能开通每种业务一次。目前 $cs1$ 开通了 $sv1, sv2$ 两种业务，$cs2$ 开通了 $sv3$ 一种业务；假设开通动作和取消动作只有客户与所开通的业务两种性质；任意一个客户和任意一种业务都可以组成一个开通与取消的动作。现在考虑 $cs1$ 开通 $sv4$ 和 $cs1$ 取消 $sv2$ 两个动作个体，分别用 d_1，d_2 表示。

另外，假设一个投诉动作，用来表示用户对业务满意程度的反应。假设投诉动作有客户和所开通的业务两种性质，考虑 $cs1$ 投诉 $svr1$ 这个动作，用 d_3 表示，它并不改变业务模型的状态。引进一个更新动作，用来表示用户对业务的更新操作，用 d_4 表示，考虑 $cs1$ 更新 $svr4$ 这个动作。

3.2 通用业务的动态概念模型的语法

本节分别给出动作层、中间层、对象层不同符号的定义。采用描述逻辑与动态命题逻辑相结合来表示上述通用业务模型。

定义动作层的符号如下。

(1)动作概念：$Deploy, Cancel, Complaint, Replace$；

(2)动作角色：$DeployedBy, OnDeploy, CanceledBy, OnCanceled, ComplainedBy, OnComplain, ReplacedBy, OnReplace$；

(3)原子动作：$\alpha_1, \alpha_2, \alpha_3, \alpha_4$；

(4)构造子：$=1$。

定义中间层的符号如下。

(1)原子动作模态词：$[\alpha_1], [\alpha_2], [\alpha_3], [\alpha_4]$。

定义对象层的符号如下。

(1)对象概念：$Service, Customer, Server, TransactedService, UpdatedService, CostPerformance$；

(2)对象角色：$Transact, ApproveTo, Complain, Update$；

(3)客户个体：$customer1, customer2, customer3$；

(4)服务个体：$service1, service2, service3, service4, service5$；

(5)其他个体：$server1, quality1$；

(6)概念构造子：$\sqcap, \rightharpoonup, \bot, \exists, \forall, T$

(7)角色构造子：$^-$；

(8)布尔符号：$\wedge, \rightharpoonup, \rightarrow$。

其中，动作层中的 α_1 表示 $cs1$ 开通 $sv4$ 这个动作个体，α_2 表示 $cs1$ 取消 $sv2$ 这个动作个体，α_3 表示 $cs1$ 投诉 $svr1$ 这个动作个体，α_4 表示 $cs1$ 更新 $sv4$ 这个动作。中间层中的 $[\alpha_1]$，$[\alpha_2]$，$[\alpha_3]$，$[\alpha_4]$ 表示相应的模态词，它们作用在对象层的概念和公式上，形成动态概念和动态公式，比如 $[\alpha_1]TransactedService$，$[\alpha_2]Cancel(cs1, sv2)$ 和 $[\alpha_3]Complain(cs1, svr1)$。

3.3 业务模型中动态概念模型的语义

通用业务模型的表示分为动作层、对象层及中间层。中间层的模型是一个动态模型，是对象层的动态变化模型，所以将对象层的模型包含到动作层的模型中，则业务模型的语义模型由动作层和中间层组成。用 M 表示通用业务的语义模型，M_1 表示动作层的模型，M_2 表示中间层的模型，则 M 由二元组组成：$M=(M_1, M_2)$。

M_1 是一个描述逻辑模型，$M_1=(\sum, I_1)$。其中 $\sum=\{d_1, d_2, d_3, d_4\}$；$I_1$ 是一个解释函数，将动作概念解释到 $\sum$ 上的子集，将动作角色解释到从 $\sum$ 到 M_2 中的论域上的二元关系。

M_2 是一个动态描述逻辑模型，$M_2=(W, T_{a1}, T_{a2}, T_{a4}, I)$。其中 $W=\{w_1, w_2, w_3, w_4, w_5, w_6\}$；$T_{a1}=\{(w_1, w_2), (w_3, w_4)\}$；$T_{a2}=\{(w_1, w_3), (w_2, w_4)\}$；$T_{a4}=\{(w_2, w_5)(w_4, w_6)\}$；$I$ 是一个解释函数，将 W 中每个状态映射到一个描述逻辑模型，每个状态映射到相同的论域 $\Delta=\{sv1, sv2, sv3, sv4, sv5, cs1, cs2, cs3, svr1, qual1\}$。

I_1 对动作个体名称、动作概念和动作角色的解释如下：

$\alpha_1^{I_1}=d_1$

$\alpha_2^{I_1}=d_2$

$\alpha_3^{I_1}=d_3$

$\alpha_4^{I_1}=d_4$

$Deploy^{I_1}=\{d_1\}$

$Cancel^{I_1}=\{d_2\}$

$Complaint^{I_1}=\{d_3\}$

$Replace^{I_1}=\{d_4\}$

$ApproveTo^{I_1}=\{(d_1,cs1)\}$

$OnDeploy^{I_1}=\{(d_1,sv4)\}$

$CanceledBy^{I_1}=\{(d_2,cs1)\}$

$OnCancel^{I_1}=\{(d_2,sv2)\}$

$ComplainedBy^{I_1}=\{(d_3,cs1)\}$

$OnComplain^{I_1}=\{(d_3,svr1)\}$

$ReplacedBy^{I_1}=\{(d_4,cs1)\}$

$OnReplace^{I_1}=\{(d_4,sv4)\}$

I 将 w_1 映射到如下描述逻辑模型：

$I(w_1)=(\Delta,\cdot^{I,w_1})$

$customer1^{I,w_1}=cs1,\cdots$

$service1^{I,w_1}=sv1,\cdots$

$server1^{I,w_1}=svr1$

$Service^{I,w_1}=\{sv1,sv2,sv3,sv4\}$

$Customer^{I,w_1}=\{cs1,cs2,cs3\}$

$Server^{I,w_1}=\{svr1\}$

$TransactedService^{I,w_1}=\{sv1,sv2,sv3\}$

$Transact^{I,w_1}=\{(cs1,sv1),(cs1,sv2),(cs2,sv3)\}$

I 将 w_2 映射到如下描述逻辑模型：

$I(w_2)=(\Delta,\cdot^{I,w_2})$

$customer1^{I,w_2}=cs1,\cdots$

$service1^{I,w_2}=sv1,\cdots$

$server1^{I,w_2}=svr1$

$quality1^{I,w_2}=qual1$

$Service^{I,w_2}=\{sv1,sv2,sv3,sv4\}$

$Customer^{I,w_2}=\{cs1,cs2,cs3\}$

$Server^{I,w_2}=\{svr1\}$

$CostPerformance^{I,w_2}=\{qual1\}$

$TransactedService^{I,w_2}=\{sv1,sv2,sv3,sv4\}$

$Transact^{I,w_2}=\{(cs1,sv1),(cs1,sv2),(cs2,sv3),(cs1,sv4)\}$

$Complain^{I,w_2}=\{(cs1,svr1)\}$

I 将 w_3 映射到如下描述逻辑模型：

$I(w_3)=(\Delta,\cdot^{I,w_3})$

$customer1^{I,w_3}=cs1,\cdots$

$service1^{I,w_3}=sv1,\cdots$

$server1^{I,w_3}=svr1$

$Service^{I,w_3}=\{sv1,sv2,sv3,sv4\}$

$Customer^{I,w_3}=\{cs1,cs2,cs3\}$

$Server^{I,w_3}=\{svr1\}$

$TransactedService^{I,w_3}=\{sv1,sv3\}$

$Transact^{I,w_3}=\{(cs1,sv1),(cs2,sv3)\}$

I 将 w_4 映射到如下描述逻辑模型：

$I(w_4)=(\Delta,\cdot^{I,w_4})$

$customer1^{I,w_4}=cs1,\cdots$

$service1^{I,w_4}=sv1,\cdots$

$server1^{I,w_4}=svr1$

$quality1^{I,w_4}=qual1$

$Service^{I,w_4}=\{sv1,sv2,sv3,sv4\}$

$Customer^{I,w_4}=\{cs1,cs2,cs3\}$

$Server^{I,w_4}=\{svr1\}$

$CostPerformance^{I,w_4}=\{qual1\}$

$TransactedService^{I,w_4}=\{sv1,sv3,sv4\}$

$Transact^{I,w_4}=\{(cs1,sv1),(cs2,sv3),(cs1,sv4)\}$

$Complain^{I,w_4}=\{(cs1,svr1)\}$

I 将 w_5 映射到如下描述逻辑模型：

$I(w_5)=(\Delta,\cdot^{I,w_5})$

$customer1^{I,w_5}=cs1,\cdots$

$service1^{I,w_5}=sv1,\cdots$

$Service^{I,w_5}=\{sv1,sv2,sv3,sv4\}$

$Customer^{I,w_5}=\{cs1,cs2,cs3\}$

$TransactedService^{I,w_5}=\{sv1,sv2,sv3,sv5\}$

$Transact^{I,w_5}=\{(cs1,sv1),(cs1,sv2),(cs2,sv3),(cs1,sv5)\}$

$UpdatedService^{I,w_5}=\{sv4\}$

$Update^{I,w_5}=\{(cs1,sv4)\}$

I 将 w_6 映射到如下描述逻辑模型：

$I(w_6)=(\Delta,\cdot^{I,w_6})$

$customer1^{I,w_6}=cs1,\cdots$

$service1^{I,w_6}=sv1,\cdots$

$Service^{I,w_6}=\{sv1,sv2,sv3,sv5\}$

$Customer^{I,w_6}=\{cs1,cs2,cs3\}$

$TransactedService^{I,w_6}=\{sv1,sv3,sv5\}$

$Transact^{I,w_6}=\{(cs1,sv1),(cs2,sv3),(cs1,sv5)\}$

$UpdatedService^{I,w_6}=\{sv4\}$

$Update^{I,w_6}=\{(cs1,sv4)\}$

可以看到，概念 *Service* 和 *Customer* 的解释不随动作的变化而变化，而 *TransactedService* 和 *Transact* 的解释随动作的变化而变化。图 2 给出了各种动作与状态之间的转换关系。

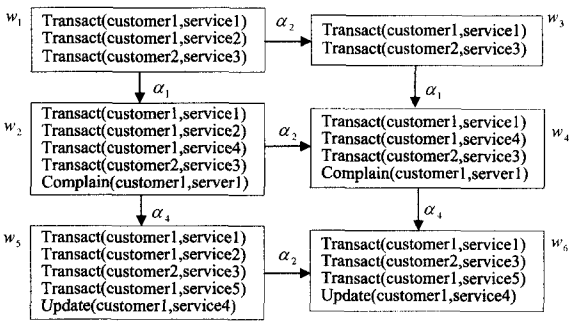

图 2　状态与动作之间的转换关系

3.4　动作规则的表示

从上述模型可以表示出通用业务模型中的如下规则：

(1)模型处于 w_1 状态，即 *sv*4 没有被开通的状态下，发生动作 d_1 后，模型状态转换到 w_2，*sv*4 被开通了：

$\neg TransactedService(service4)\rightarrow[\alpha_1]TransactedService(service4)$

(2)在 w_2 中，有客户 *cs*1 投诉了服务商 *svr*1。投诉动作的发生有约束条件：*Customer*1 开通了 *service*4，且 *service*4 的质量为 *quality*1。满足这两个条件的情况下，发生动作 d_3，客户 *cs*1 与服务商 *svr*1 具有了投诉与被投诉的关系。但是，由于动作对模型状态的影响主要体现在客户与业务之间的关系上，因此动作 d_3 的发生并不会对模型的状态发生影响，即在

w_2 状态下，发生 d_3 动作，模型不会到达任何另外的状态。规则如下：

$Transact(customer1, service4) \wedge CostPerformance(quality1) \rightarrow [\alpha_3]Complain(customer1, server1), Transact(customer1, service4) \rightarrow ([\alpha_3]Complain(customer1, server1)) \equiv T$

(3)w_2 状态下，$cs1$ 开通了 $sv4$，或者 $sv4$ 处于被开通状态。在这个条件下发生动作 d_4 和 $sv4$ 被更新成为 $sv5$，即 $sv5$ 被开通，$sv4$ 不再处于开通状态：

$TransactedService(service4) \rightarrow [\alpha_4]TransactedService(service5)$

$TransactedService(service4) \rightarrow [\alpha_4]UpdatedService(service4)$

$Transact(customer1, service4) \rightarrow [\alpha_4] \rightarrow TransactedService(service4)$

w_1 状态下，$cs1$ 没有开通 $sv4$，发生动作 d_1，d_4 后，$sv5$ 被开通了：

$\rightarrow Transact(customer1, service4) \rightarrow [\alpha_1; \alpha_4]TransactedService(service5)$

在 $sv4$ 没有被更新的状态下，发生动作 d_4 后，$sv4$ 被更新，且 $sv5$ 被开通了：

$\rightarrow UpdatedService(service4) \rightarrow [\alpha_4]UpdatedService(service4)$

$\rightarrow UpdatedService(service4) \rightarrow [\alpha_4]TransactedService(service5)$

(4)在 $sv4$ 已经被开通的状态下，发生动作 d_1 不会到达任何一个状态：

$TransactedService(service4) \rightarrow ([\alpha_1]TransactedService) \equiv T$

(5)在 $sv2$ 已经被开通，或者 $cs1$ 已经开通过 $sv2$ 的状态下，发生动作 d_2 后，$sv2$ 不再处于开通状态，或者说 $cs1$ 与 $sv2$ 不再有开通与被开通的关系：

$TransactedService(service2) \rightarrow [\alpha_2] \rightarrow TransactedService(service2)$

$Transact(customer1, service2) \rightarrow [\alpha_2] \rightarrow Transact(customer1, service2)$

(6)在 $sv2$ 没有被开通的状态下，发生动作 d_2 不会到达任何一个状态：

$\rightarrow Transact(customer1, service2) \rightarrow ([\alpha_2]TransactedService) \equiv T$

$\rightarrow TransactedService(service2) \rightarrow ([\alpha_2]TransactedService) \equiv T$

4 服务业务 DDL 模型上的推理

4.1 DDL 推理方法

定义 1(一致性)[9,10] 一个断言公式集 $\mathcal{F}$ 是一致的，当且仅当 $\mathcal{F}$ 中不含冲突，否则称 $\mathcal{F}$ 是不一致的。给定一个 TBox $\mathcal{T}$，如果 $\mathcal{F}$ 在 $\mathcal{T}$ 中是可满足的，则称 $\mathcal{F}$ 关于 $\mathcal{T}$ 一致。

定义 2(原子动作的一致性)[9,10] 给定 $\mathcal{T}$ 和原子动作 $A(x_1, \cdots, x_m) = (P_A, E_A)$，对于任何一个替换 θ，

1)如果由 $P_A\theta$ 组成的断言公式集 $\mathcal{F}$ 关于 $\mathcal{T}$ 一致，则动作 A 是前提一致的；

2)通过下列步骤得到断言公式集 $\mathcal{F}$：

(a)令 $\mathcal{F}=\phi$；

(b)对于 $\forall B_i/H_i \in E_A$，若 $\mathcal{T} \models B_i\theta$，则 $\mathcal{F}=\mathcal{F} \cup H_i\theta$；

如果 $\mathcal{F}$ 关于 $\mathcal{T}$ 一致，则称动作 A 是一致的。

3)如果 A 是前提一致且结果一致的，则称动作 A 是一致的。

其中 P_A 是动作的前提条件，E_A 是动作的执行结果，B_i/H_i 是 E_A 的表现形式。

定义 3(复杂动作的一致性)[9] 复杂动作一致，当且仅当组成它的每一个原子动作一致。

下面给出原子动作一致性检验算法。

算法 1(动作一致性检测算法)[8,10] 给定知识库 $\mathcal{K}=(\mathcal{T}, \mathcal{A})$，原子动作 $A(x_1, \cdots, x_m)=(P_A, E_A)$ 和替换 $\theta=(x_1/a_1, \cdots, x_m/a_m)$

1)令 $\Gamma=\mathcal{A} \cup P_A\theta$，构造 $\Gamma \cup \mathcal{T}$ 的初始约束系统，记为 S_Γ；

2)针对 S_Γ 反复应用传播规则，直到没有规则可以应用为止，得到 S_Γ 的完全扩展树，记为 $\mathcal{S}$；

3)对于 $\mathcal{S}$ 每条没有冲突的分支 $S \in \mathcal{S}$，构造解释 I_s，

(a)令 $\Sigma=\emptyset$；

(b)对于 E_A 中每个条件表达式 B_i/H_i，如果 $I_s \models B_i\theta$，则 $\Sigma=\Sigma \cup H_i\theta$；

(c)令 $\Sigma=\Sigma \cup \mathcal{T}$，构造 Σ 的初始约束系统 S_Σ；

(d)针对 S_Σ 反复应用传播规则，直到没有规则可以应用为止，得到 S_Σ 的完全扩展树，若树中存在没有冲突的分支，则动作 A 在解释 I_s 中是一致的。

该算法利用 *Tableaux* 算法[11]检验动作的前提条件与执行结果，其可用于检测原子动作的一致性。

算法 2(构造初始约束系统算法)[9] 给定公式集 Π 和 TBox $\mathcal{T}$，关于 $\Pi \cup \mathcal{T}$ 的初始构造系统 S_Π 是通过以下方式构造的：

1)对于公式集 Π 中具有形式 $C(a)$ 的断言公式，将 $a:C$ 加入 S_Π；

2)对于公式集 Π 中具有形式 $R(a,b)$ 的断言公式，如果 R 是原子角色，将 aRb 加入 S_Π；如果 $R \equiv P_1 \cap \cdots \cap P_k$，其中 P_i 是原子角色，将 $aP_1b, \cdots, aP_kb$ 加入 S_Π；

3)对于 $\mathcal{T}$ 每一个包含声明 $C \sqsubseteq D$，将 $\forall x . x: \neg C \cup D$ 加入 S_Π；

4)对于 Π 中不同名称的任意两对个体 a 和 b，将 $a \neq b$ 加入 S_Π。

4.2 DDL 推理举例

给定知识库 $\mathcal{K}=(\mathcal{T}, \mathcal{A})$，其中 $\mathcal{T}$ 描述系统的语义内容，通过数据库查询的信息存储在 $\mathcal{A}$ 中，$\mathcal{T}$ 包括：概念集 $=\{Service, Department, Flag, Manager, Server\}$、角色集 $=\{manage, workIn, flag, canSearch, canTransact\}$、公理集 $=\{manage \in workIn, Flag = \{forbidden, searchtransact, searchonly\}, Manager \equiv Server \cap \exists manager.Department\}$。

假定规则为：若业务标志为 *forbidden*，则只有本部门员工可以查询业务，但只有部门经理可以开通该业务；若标志为 *serchonly*，则任何人可以查询该业务，但只有部门经理可以开通该业务；若标志为 *searchtransact*，则任何人可以查询该业务，但只有部门员工可以开通该业务。

定义一组简单动作如下：

$GetFlag(x,y) = (\{Service(x)\}, \{Flag(y), flag(x,y)\})$

$GetSeDep(x,y) = (\{Server(x)\}, \{Department(y), wor-$

$kIn(x,y)\}$)

$GetSerDep(x,y)=(\{Service(x)\},\{Department(y),belongTo(x,y)\})$

$SetSearch(x,y)=(\{Server(x),Service(y)\},\{canSearch(x,y)\})$

$SetTransact(x,y)=(\{Server(x),Service(y)\},\{canTransact(x,y)\})$

$Check(x,y,z)=(\{Server(x),Service(y)\},\{Department(z),workIn(x,z),belongTo(y,z)\}/canSearch(x,y),\{Department(z),workIn(x,z),belongTo(y,z)\}/canTransact(x,y)\})$

其中，

$GetFlag(x,y)$：查询获得业务 x 的标志 y；

$GetEmDep(x,y)$：查询获得员工 x 所在的部门 y；

$GetSerDep(x,y)$：查询获得业务 x 的所属部门 y；

$SetSearch(x,y)$：设置 x 可以查询业务 y；

$SetTransact(x,y)$：设置 x 可以开通业务 y；

$Check(x,y,z)$：查询 x 是否有权限查询、开通 z 部门的业务 y。

基于上述简单动作，复杂动作定义如下：

$Search(x,y,z,w,u)=GetFlag(y,w)$

$((flag(y,searchonly)\vee flag(y,searchtransact))?$

$SetSearch(x,y)\cup Flag(y,forbidden)?$

$GetSeDep(x,z)$

$GetSerDep(y,u)$

$Check(x,y,z))$

$Search(x,y,z,w,u)$表示先获得业务 y 的标志 w，如果标志不为 $forbidden$，则 x 可以查看业务 y，否则需要查看部门 z 的服务人员 x 是否有权限查看部门 u 的业务 y。

下面讨论动作 $Search(x,y,z,w,u)$的执行。

给定 $\mathcal{A}=\{Manager(ID1),Service(svc1)\}$和 $\theta=(x/ID1,y/svc1)$，则 $Search(ID1,svc1,z,w,u)=Search(x,y,z,w,u)\theta$ 执行如下：

1)$GetFlag(x,y)$动作可执行，查询得到 $svc1$ 的标志，在 A 中加入得到的事实 $flag(svc1,forbidden)$，即 $\mathcal{A}=\mathcal{A}\cup\{flag(svc1,forbidden)\}$，且 $\theta=(x/ID1,y/svc1,w/forbidden)$；

2)由于 $\mathcal{T},\mathcal{A}\models flag(svc1,searchonly)\vee flag(svc1,searchtransact)$，故 $flag(y,searchonly)\vee flag(y,searchtransact)?$；$SetSearch(x,y)$不可执行；

3)由于 $\mathcal{T},\mathcal{A}\vdash flag(svc1,forbidden)$，执行：

(a)查询 $ID1$ 所属的部门，得到 $\mathcal{A}=\mathcal{A}\cup\{manage(ID1,CreditCardCenter)\}$，$\theta=(x/ID1,y/svc1,z/CreditCardCenter,w/forbidden)$；

(b)查询业务 $svc1$ 所属的部门，得到 $\mathcal{A}=\mathcal{A}\cup\{belongTo(svc1,CreditCardCenter)\}$，$\theta=(x/ID1,y/svc1,z/CreditCardCenter,w/forbidden,u/CreditCardCenter)$；

(c)从 $manage(ID1,CreditCardCenter)$推理得到 $workIn(ID1,CreditCardCenter)$，$\mathcal{A}=\mathcal{A}\cup\{workIn(ID1,CreditCardCenter)\}$，执行 $Check(ID1,svc1,CreditCardCenter)$，得到 $canSearch(ID1,svc1)$，$canTransact(ID1,svc1)$，$\mathcal{A}=\mathcal{A}\cup\{canSearch(ID1,svc1),canTransact(ID1,svc1)\}$。

下面讨论动作 $Check(x,y,z)$的一致性。给定：

$\mathcal{A}=\{Manager(ID1),Service(svc1),Department(CreditCardCenter),manage(ID1,CreditCardCenter),belongTo(svc1,CreditCardCenter)\}$

$\theta=(x/ID1,y/svc1,z/CreditCartCenter)$。

为方便表示，采用表 1 所列的符号表示动作、概念、角色、个体。

表 1　符号表

符号	名称	符号	名称
A	动作 Check(x,y,z)	P_1	角色 manage
E	概念 Server	P_2	角色 workIn
D	概念 Department	P_3	角色 belongTo
M	概念 Manager	P_4	角色 canSearch
S	概念 Service	P_5	角色 canTransact
m	个体 ID1	c	个体 CreditCardCenter
s	个体 svc1	…	…

得到：

$P_A\theta=\{E(m),S(s)\}$

$E_A\theta=\{\{D(c),P_2(m,c),P_3(s,c)\}/P_4(m,s),\{D(c),P_2(m,c),P_3(s,c)\}/P_5(m,s)\}$

$\Gamma=\{M(m),S(s),D(c),P_1(m,c),P_3(s,c)\}$

$S_\Gamma=\{m:M,s:S,c:D,mP_1c,sP_3c,M\equiv E\sqcap\exists P_1.D,P_1\subseteq P_2,m\neq s,m\neq c,s\neq c,\cdots\}$

可以推理得到：$S=S_\Gamma\cup\{m:E,m:\exists P_1.D,mP_2c,\cdots\}$。

由 S 一致、$P_A\theta\subset S$ 可见，$P_A\theta$ 是可满足的，A 在$(\mathcal{T},\mathcal{A})$中前提一致。

接着查看 $E_A\theta$，对于$\{D(c),P_2(m,c),P_3(s,c)\}/P_4(m,s)$和$\{D(c),P_2(m,c),P_3(s,c)\}/P_5(m,s)$，得到：

$\Sigma=\{P_4(m,s),P_5(m,s)\}$

$S_\Sigma=\{mP_4s,mP_5s,M\equiv E\sqcap\exists P_1.D,P_1\subseteq P_2,\cdots\}$

可以得到 S_Σ 是可满足的，A 在 $\mathcal{T}$ 中是结果一致的。

由前提一致和结果一致得到动作 $Check(x,y,z)$是一致的。

结束语　本文在参照都柏林核心的基础上，建立了适用于服务业的通用业务模型。提出了主题概念，并对主题概念分层，从结构上比较清晰地描述了服务业的知识资源。为了更清楚地表示模型，采用动态描述逻辑，并引入三层结构思想，对模型中的动作、业务、用户对象等进行描述。使用三层结构建立的逻辑模型可以很好地表示开通、取消等动作的动态过程，还可以描述模型所处的状态。

根据上述研究，我们认为，利用元数据建模与动态描述逻辑描述的方法，可以有效建立统一的、严格的业务动态描述方法和模型。

参 考 文 献

[1] Newman D, Hagedorn K, Chemudugunta C, et al. Subject metadata enrichment using statistical topic models[M]. ACM, 2007: 366-375

[2] Margaritopoulos T, Margaritopoulos M, Mavridis I, et al. A Conceptual Framework for Metadata Quality Assessment[J]. International Journal of Metadata, Semantics and Ontologies, 2008, 3(4): 292-304

[3] DCMI. Dublin Core Metadata Element Set [EB/OL]. http://dublincore.org/docu ments/Dces

[4] 从晓青，曹存根，眭跃飞. 带函数的描述逻辑[J]. 计算机工程与应用，2008，44(22)：46-50

[5] 丛晓青. 一种动态描述逻辑及其应用[D]. 北京：中国科学院研究生院，2008
[6] Description Logic Website[OL]. http://dl.kr.org/
[7] Baader F, Nutt W. Basic Description Logics[M]. Cambridge University Press, 2003: 47-100
[8] Wolter F, Zakharyaschev M. Dynamic description logics [R]. CSLI Publications, 2000
[9] 黄河. 语义 Web 中知识服务的研究[D]. 北京：中国科学院研究生院，2006
[10] 董明楷. 面向智能主体的动态描述逻辑研究[D]. 北京：中国科学院研究生院，2003
[11] Baader F, Sattler U. An Overview of Tableau Algorithms for Description Logics[J]. Studia Logica, 2001, 69: 5-40
[12] Chau K W. An ontology-based knowledge management system for flow and water quality modeling[J]. Advances in Engineering Software, 2007, 38: 172-781
[13] Alex B, Ronald J B. Conceptual modeling with description logics [M]. Cambridge University Press, 2003: 359-381
[14] 吴修国，曾广周，许崇敬. 基于描述逻辑的目标研究[J]. 计算机科学，2008，35(7)：142-144

(上接第 227 页)

方面，BDE-EDA 算法和 TCNN-HS 算法在 5 个问题实例上均能获得与已知最优解相同的平均解，而在其他的 11 个问题实例上 BDE-EDA 算法能获得比 TCNN-HS 算法更好的平均解，BDE 和 TCNN 在所有问题实例上获得的平均解均比 BDE-EDA 的差。可以说 BDE-EDA 算法具有比 BDE 和 TCHNN 算法更好的性能，并且具有比 TCNN-HS 算法更稳定的性能。表 1 中各算法求得的与已知最优解相同的最好解和平均解用加粗字体标出。

从表 2 可以看出，BDE-EDA 算法的运行时间比 BDE 稍长，这是因为 BDE-EDA 算法要花费时间向优质个体学习建立概率模型，但是二者的时间差距不是很明显。TCNN-HS 算法的运行时间最长，而且随着问题规模的增加算法的速度变得越来越慢。

综上所述，从解的质量和运行时间来看，BDE-EDA 的运行虽然略慢于 BDE，但是它的最优解和平均解的质量明显比 BDE 要好。而 TCNN-HS 算法虽然能取得与 BDE-EDA 相当的最优解，但是在稳定性以及运行时间上比不上 BDE-EDA 算法。因此，可以说 BDE-EDA 算法是一个更稳定、有更好寻优能力且运行效率较高的算法。

结束语 提出了具有学习机制的离散差分演化算法，并用于求解多维背包问题，实验结果表明提出的算法具有良好的性能。未来的研究工作是将该算法与其他机制相结合来进一步提高性能，例如结合模拟退火等算法，并将算法推广到实际应用中。

参 考 文 献

[1] Price K. Differential Evolution V S. The Functions of the 2nd ICEO[C]// Proceedings of IEEE International Conference on Evolutionary Computation. Indianapolis, USA, 1997: 153-157
[2] Storn R, Price K. Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces [R]. International Computer Science Institute, Berkley, 1995
[3] 陈荣元，林立宇，王四春，等. 数据同化框架下基于差分进化的遥感图像融合[J]. 自动化学报，2010，36(3)：392-398
[4] 王凌，黄付卓，李灵坡. 基于混合双种群差分进化的电力系统经济负荷分配[J]. 控制与决策，2009，24(8)：1156-1160
[5] 李太勇，唐常杰，吴江，等. 基于差分进化基因表达式编程的全局函数优化[J]. 计算机科学，2009，36(11)：140-142
[6] 周树德，孙增圻. 分布估计算法综述[J]. 自动化学报，2007，33(2)：113-124
[7] Müehlenbein H. The equation for response to selection and its use for prediction[J]. Evolutionary Computation, 1997, 5(3): 303-346
[8] Baluja S. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning[R]. CMU-CS-94-163. School of Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, 1994
[9] 胡蓉，钱斌. 一种求解随机有限缓冲区流水线调度的混合差分进化算法[J]. 自动化学报，2009，35(12)：1580-1586
[10] 吴亮红，王耀南，陈正龙. 求解混合整数非线性规划问题的改进差分进化算法[J]. 小型微型计算机系统，2007，28(4)：666-669
[11] Qian B, Wang L, Huang D X, Wang X. Scheduling multi-objective job shops using a memetic algorithm based on differential evolution[J]. International Journal of Advanced Manufacturing Technology, 2008, 35(9/10): 1014-1027
[12] Al-Anzi F S, Allahverdi A. A self-adaptive differential evolution heuristic for two-stage assembly scheduling problem to minimize maximum lateness with setup times[J]. European Journal of Operational Research, 2007, 182: 80-94
[13] Engelbrecht A P, Pampara G. Binary differential evolution strategies[C]// IEEE Congress on evolutionary computation. 2007: 1942-1947
[14] Onwubolu G, Davendra D. Scheduling flow shops using differential evolution algorithm[J]. European Journal of Operational Research, 2006, 171: 674-692
[15] Gong T, Tuson A L. Differential evolution for binary encoding [J]. Soft Computing in Industrial Applications, 2007, ASC 39: 251-262
[16] Chang P C, Chen S H, Fan C Y, et al. Genetic algorithm integrated with artificial chromosomes for multi-objective flowshop scheduling problems[J]. Applied Mathhematics and Computation, 2008, 205: 550-561
[17] Wang B Y, Dong H, He Z Y. A chaotic annealing neural network with gain sharpening and its application to the 0/1 knapsack problem[J]. Neural Processing Letters, 1999: 243-247
[18] Zhou Y, Kuang Z H, Wang J H. A chaotic neural network combined heuristic strategy for multidimensional knapsack problem [C]// ISICA 2008, LNCS 5370. 2008: 715-722
[19] Kong M, Tian P, Kao Y. A new ant colony optimization algorithm for the multidimensional knapsack problem[J]. Computers and Operations Research, 2008, 35(8): 2672-2683
[20] 徐青鹤，刘士荣，吕强. 基于蚁群混沌行为的离散粒子群算法及其应用[J]. 计算机科学，2010，37(5)：178-180
[21] 曾智，杨小帆，陈静，等. 求解多维 0-1 背包问题的一种改进的遗传算法[J]. 计算机科学，2006，33(7)：220-223
[22] Li H, Jiao Y C, Zhang L, et al. Genetic algorithm based on the orthogonal design for multidimensional knapsack problems[C]// ICNC 2006. LNCS. Vol. 4221, Springer, Heidelberg, 2006: 696-705