软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2018,29(6):1647−1669 [doi: 10.13328/j.cnki.jos.005460]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于类型理论的领域数据建模和验证及案例*

乌尼日其其格, 李小平, 马世龙, 吕江花

(软件开发环境国家重点实验室(北京航空航天大学),北京 100083)

通讯作者: 吕江花, E-mail:jhlv@nlsde.buaa.edu.cn

摘 要: 数据作为软件系统的主要处理对象,其规范性有助于软件系统的设计开发和软件系统之间的数据交换.面向行业数据规范及其验证,提出了一种基于类型理论的领域数据建模语言(DDML)和领域建模方法(DDMM).DDML 语言通过定义类型和项的语法和语义,描述领域数据类型和对象的结构,通过定义类型规则及其类型检查算法判定任意项 $t:T$?.DDMM 给出了领域数据建模的方法,即构建$\mathcal{K}_1$(原子类型)、$\mathcal{K}_2$(数据元)、$\mathcal{K}_3$(数据元目录)三层框架,生成表示$\mathcal{K}_3$ 层数据元目录之间关系的类型规则.在此基础上,给出了数据元目录序列的定义及其正确性判定算法.基于上述方法,实现了一种领域数据建模工具原型系统,并通过领域数据建模与自动验证的一个实际案例,完成了一个较大规模行业数据规范的制定与验证.

关键词: 类型理论;类型检查;类型规则;领域数据建模;数据规范

中图法分类号: TP311

中文引用格式: 乌尼日其其格,李小平,马世龙,吕江花.基于类型理论的领域数据建模和验证及案例.软件学报,2018,29(6):1647−1669. http://www.jos.org.cn/1000-9825/5460.htm

英文引用格式: Wuniri QQG, Li XP, Ma SL, Lü JH. Type theory based domain data modelling and verification with case study. Ruan Jian Xue Bao/Journal of Software, 2018,29(6):1647−1669 (in Chinese). http://www.jos.org.cn/1000-9825/5460.htm

Type Theory Based Domain Data Modelling and Verification with Case Study

WUNIRI Qi-Qi-Ge, LI Xiao-Ping, MA Shi-Long, LÜ Jiang-Hua

(State Key Laboratory of Software Development Environment (Beihang University), Beijing 100083, China)

Abstract: As the main object manipulated by a software system, data with a domain standard can contribute to the process of software design and data shareware between software systems. In this paper, focusing on domain data standardization, a domain data modelling language (DDML) and a domain data modelling method (DDMM based on the type theory are proposed. In DDML, the syntax and semantics of types and terms are defined to describe the structure of the domain data types and objects, and the typing rules are defined to process the judgement of $t:T$. For DDMM, the data modelling processes are presented with the data modelling of $\mathcal{K}_1$ (atomic type), $\mathcal{K}_2$ (data element), $\mathcal{K}_3$ (data element directory) and the generation of typing rules in $\mathcal{K}_3$. Furthermore, the definition of the data element directory sequences and the algorithms of checking its correctness are defined. Finally, a prototype of the domain data modelling system as a modelling tool is developed and applied to a real case of large scale by the generation of the domain data standard and its evaluation.

Key words: type theory; type checking; typing rules; domaindata modelling; data standard

* 基金项目: 国家自然科学基金(61003016, 61300007, 61305054); 科技部基本科研业务费重点科技创新类项目(YWF-14-JSJXY-007); 软件开发环境国家重点实验室自主探索基金(SKLSDE-2012ZX-28, SKLSDE-2014ZX-06)

Foundation item: National Natural Science Foundation of China (61003016, 61300007, 61305054); Base Research Foundation of Ministry of Science and Technology of China (YWF-14-JSJXY-007); Independent Discovery Foundation of State Key Laboratory of Software Development Environment of China (SKLSDE-2012ZX-28, SKLSDE-2014ZX-06)

本文由形式化方法的理论基础专题特约编辑傅育熙教授、李国强副教授、田聪教授推荐.

收稿时间: 2017-06-27; 修改时间: 2017-09-01; 采用时间: 2017-11-06; jos 在线出版时间: 2017-12-28

CNKI 网络优先出版: 2017-12-29 13:19:03, http://kns.cnki.net/kcms/detail/11.2560.TP.20171229.1318.002.html

领域数据建模是制定和验证行业数据规范的基础.数据规范的制定,对行业信息化建设起到指导作用,对行业数据的分析和基于数据分析的决策至关重要.大规模分布式复杂应用系统是目前应用系统的主流,随着应用系统规模的逐渐增大,行业数据规范制定越来越复杂,特别是数据规范编写满足需求的验证也变得复杂和困难.传统的人工编写方法难以应对这种复杂性,研发行业数据规范形式化编写方法和自动化验证方法,成为今后必然的发展趋势.

作为一种轻量级形式化方法,类型系统研究始于 20 世纪初,到 70 年代已成为证明论中的标准工具,也是程序设计语言的基础[1].类型理论通过类型、求值和类型规则的定义和解释,研究项的可类型化和项的良类型等性质.类型系统通过静态检查可以检查错误,通过抽象还可以强化规范编程.一个软件系统中出现的任何一个数据或程序片段都可以认为是项,因此,类型理论可以作为领域数据建模的理论基础,通过构造一个类型系统,可以进行领域数据建模并进行验证.

本文面向行业数据规范及其验证,通过分析应用需求和行业数据特点,提出了一种基于类型理论的领域数据建模语言(domain data modelling language,简称 DDML)和领域数据建模方法(domain data modelling method,简称 DDMM),并将其应用于实际规模的应用案例中,说明了该方法的有效性.为此,本文第 1 节综述类型理论的研究现状.第 2 节的 DDML 语言定义了类型和项的语法和语义,以及类型规则及其类型检查算法.DDMM 方法则给出了领域数据建模的步骤,即构建$\mathcal{K}_1$(原子类型)、$\mathcal{K}_2$(数据元)、$\mathcal{K}_3$(数据元目录)3 层框架,以及如何生成表示$\mathcal{K}_3$层数据元目录之间关系的类型规则.在此基础上,给出数据元目录序列的定义及其正确性判定算法.第 3 节面向某行业实际数据规范的制定与验证,通过需求分析,采用领域数据建模工具原型系统进行数据建模和验证.最后一节给出本文在理论和应用上的主要贡献.

1 相关研究

程序设计语言的设计是类型理论中重要的应用领域[2],从 FORTRAN 语言中引入类型开始[3],类型理论成为了结构化程序设计语言[4]、面向对象程序设计语言[5]以及函数式程序设计语言[6]的理论基础.此外,类型系统也用于静态分析程序正确性[7]、分析移动计算运行时的正确性[8]、程序行为分析[9]以及用于定义程序中的数据处理模式[10,11].此外,在大数据处理与分析领域,研究人员通过定义一般类型和线性类型及其类型规则,可以严格定义大数据处理中的并行计算过程,将 Map-Reduce 转化为不同类型的类型映射之和,从而可以将一个任务分而治之[12].类型理论的研究正趋于解决前沿的工程与技术问题.

领域数据建模是描述一个行业的数据以及数据之间关系的方法,是数据库、信息系统、软件工程和知识表示等领域中重要的基础工作.ER 模型是早期的数据建模方法,可以描述实体、属性和关系[13].采用 ER 建模已成为关系数据库设计与开发的起点,但该模型在关系上主要关注实体之间的主外键关系.面向对象建模方法的产物——统一建模语言(UML),在数据建模中能够描述丰富的关系,但其对数据的约束和值域的描述相对复杂[14].一种描述逻辑数据建模方法[15]扩充了 ER 模型和面向对象建模方法的描述能力,并提供了基本的推理功能,但其逻辑符号相对复杂,且更侧重于数据库中对 Schema 的一致性验证.一种多维数据建模方法主要用于联机分析处理(OLAP)中对多维复杂数据的建模[16].基于概念聚类机制(CCM)的数据模型则通过簇和角色的动态定义,主要用于视频数据库的数据建模[17].此外,类型描述语言 PCT,利用高级程序设计语言和一阶谓词逻辑可以描述数据模型中的类型、语义特征和完整性约束[18],但其数据描述能力和应用规模有限.数据建模的理论的方法众多,但是能够简洁而精确地描述一个行业数据的构造过程和关系的方法还并不多见.

类型系统作为一种轻量级形式化方法,通过类型规则可以构造复杂类型及它们之间的关系,通过类型检查算法可以判定一个给定项的类型以及给定类型之间的关系是否满足[1].为了解决应用中大规模实际问题,本文采用类型理论作为数据建模及其验证的理论基础,研究领域数据建模语言、领域数据建模方法及其验证方法,并开发相应的原型系统工具,以支持领域数据建模和自动化验证.

2 领域数据建模语言和方法

领域数据建模语言包括类型、类型语义、项、环境、类型规则、项语义这 6 个组成部分.

2.1 类 型

DDML 语言中的类型记为 X,其类型表达式定义如下:

$$X::=T|\{l:T\}|T \text{ with } \langle constraint\rangle|T^*|\{T\}|T_1\times T_2|T_1+T_2|T_1\to T_2,$$

其中,T,T_1,T_2 均为类型.

(1) 如果 l 是标签,则$\{l:T\}$也是类型;

(2) 如果 $t:T$,并且 t 满足$\langle constraint\rangle$的约束,则 T with $\langle constraint\rangle$也是类型.其中,约束 $constraint::=\tau$并且 $\tau::=\varphi|x|c|f(\tau_1,\ldots,\tau_n)|\tau_1\wedge\tau_2|\neg\tau,\varphi$为空,$x$ 是变量符号,c 是常量符号,f 是 n 元函数(或关系)符号;

(3) T^*表示类型的幂,T^*也是类型;

(4) $\{T\}$表示一个成员组成的元组类型;

(5) $T_1\times T_2$ 表示类型的聚合也是类型;

(6) T_1+T_2 表示类型的和也是类型;

(7) $T_1\to T_2$ 表示类型之间的映射也是类型.

其中,类型 $T_1\times T_2$ 也可以写为$\{T_1,T_2\}$.

对上述语法定义举例说明如下.

1) 带标签的类型$\{l:T\}$,是对类型 T 加注了标签名称,旨在突出使用用途.例如,如果有类型 *integer*,则$\{count:integer\}$也是一种类型,可以用于表示计数的整数类型;

2) 带约束的类型 T with $\langle constraint\rangle$,是对类型 T,通过增加约束,限定其可能取值.例如,对于 t:*integer*,如果 t 满足$\langle length(t)=number\rangle$约束,则 t:*integer* with $\langle length(t)=number\rangle$,即长度为 *number* 的整数;

3) 幂类型 T^*,是在类型 T 的基础上构造的类型,使用较为普遍.例如字符串类型 *string* 实为字符类型 *char* 的幂类型;

4) 单元组类型$\{T\}$,是一种特殊的元组类型,只有一个成员,其类型为 T.单元组类型在面向对象编程中相当于具有一个成员变量的类.例如,{*integer*}是单元组类型;

5) 类型的乘积 $T_1\times T_2$,是在类型 T_1 和 T_2 的基础上构造的多元元组类型,类似于面向对象编程中的封装;

6) 类型的和 T_1+T_2,是在类型 T_1 和 T_2 的基础上构造的类型,用于扩展可能取值;

7) 类型映射 $T_1\to T_2$,是一种函数类型,$\to$左端表示输入,$\to$右端表示输出.例如,*string*$\to$*integer* 可以表示输入参数为 *string*,输出参数为 *integer* 的所有函数的类型.

2.2 类型语义

设$\|T\|$表示类型 T 的值域,则 DDML 语言的类型语义定义如下.

(1) $\|\{l:T\}\|=\{\{l=a\}|a\in\|T\|\}$;

(2) $\|T \text{ with } \langle constraint\rangle\|=\{constraint(a)|a\in\|T\|\}$;

(3) $\|T^*\|=\|T\|^*$,其中,等式右上角的符号“*”表示集合幂集;

(4) $\|\{T\}\|=\{\{a\}|a\in\|T\|\}$;

(5) $\|T_1\times T_2\|=\|T_1\|\times\|T_2\|$,其中,等式右边的符号“×”表示集合的笛卡尔积;

(6) $\|T_1+T_2\|=\|T_1\|\cup\|T_2\|$,其中,等式右边的符号“∪”表示集合的并;

(7) $\|T_1\to T_2\|=\|T_1\|\to\|T_2\|$,其中,等式右边的符号“→”表示集合之间的映射.

注:由上述定义,$\|T \text{ with } \langle constraint\rangle\|$表示$\|T\|$中所有被$\langle constraint\rangle$约束的元素的集合.

对上述语义举例说明如下.

1) 带标签的类型,其语义$\|\{l:T\}\|$的值域集合,与语义$\|T\|$的值域集合相同,只是值域集合中的每个元素的形式写为$\{l=a\}$;

2) 带约束的类型,其语义$\|T \text{ with } \langle constraint\rangle\|$的值域集合,是语义$\|T\|$的值域集合的子集;

3) 幂类型,其语义$\|T^*\|$的值域集合,是语义$\|T\|$的值域集合的幂集;

4) 元组类型,其语义$\|\{T\}\|$的值域集合,是语义$\|T\|$的值域集合中每个元素加元组符号{}的集合;

5) 乘积类型,其语义$\|T_1\times T_2\|$的值域集合,是语义$\|T_1\|$的值域集合和语义$\|T_2\|$的值域集合的笛卡尔积;

6) 和类型,其语义$\|T_1+T_2\|$的值域集合,是语义$\|T_1\|$的值域集合和语义$\|T_2\|$的值域集合的并集;

7) 映射类型,其语义$\|T_1\to T_2\|$的值域集合,是语义$\|T_1\|$的值域集合到语义$\|T_2\|$的值域集合的映射.

2.3 项

DDML 语言的项 t 定义如下:

$$t::=x|c|\{l=t\}|(t_1,\ldots,t_n)|f(t_1,\ldots,t_n)|t(u)|t \text{ as } T|\text{if } t_1 \text{ then } t_2 \text{ else } t_3.$$

2.4 环 境

DDML 语言的环境Γ是变量绑定的序列,定义为

$$\Gamma::=\varnothing|\Gamma,t:T$$

为方便起见,一个非空的环境Γ经常写为 $t_1:T_1,\ldots,t_n:T_n,n\geqslant 1$.

2.5 类型规则

在环境Γ中,一个判定形为$\Gamma\vdash t:X$.通过判定该语言的类型表达式或项是否满足给定的类型规则,可以验证期望的性质.其类型规则可以分为结构类规则和关系类规则.

- 结构类规则

$$\frac{a_1:T,\ldots,a_n:T,\text{对于任意}n\geqslant 1}{a_1\ldots a_n:T^*} \quad (TR1)$$

$$\frac{x:A,x\text{满足}\langle constraint\rangle}{x:A \text{ with } \langle constraint\rangle} \quad (TR2)$$

$$\frac{t:T,l \text{ is a label}}{\{l=t\}:\{l:T\}} \quad (TR3)$$

$$\frac{a_1:T_1,\ldots,a_n:T_n,\text{对于任意}n\geqslant 1}{(a_1,\ldots,a_n):T_1\times\ldots\times T_n} \quad (TR4)$$

$$\frac{v_1:B,\ldots,v_n:B,T=\{B\}}{v_1:T,\ldots,v_n:T} \quad (TR5)$$

$$\frac{t_1:B_1,\ldots,t_n:B_n,n\geqslant 1}{range\_\exp(t_1,\ldots,t_n):B} \quad (TR6)$$

$$\frac{t=range\_\exp(t_1,\ldots,t_n):B,T=\{B\}}{t:T} \quad (TR7)$$

为下面行文方便,由类型规则(TR5)和(TR6)引入枚举类型和值域表达式类型.称 T 为枚举类型,如果 $v_1:T,\ldots,v_n:T$.由不同类型的项 $t_1:T_1,\ldots,t_n:T_n$ 组成的表达式 $range_\exp(t_1,\ldots,t_n)$表示的项的类型称为值域表达式类型.

对结构类规则(TR2)举例说明如下:

$$\frac{t:integer,t\text{ 满足}\langle length(t)=number\rangle}{t:integer \text{ with } \langle length(t)=number\rangle}.$$

为了定义关系类类型规则,首先给出如下定义.

(1) 聚合关系.如果 $T=T_1\times\ldots\times T_i\times\ldots\times T_n,1\leqslant i\leqslant n$,称类型 T_i 和类型 T 满足聚合关系,记为 $T_i \xrightarrow{\diamond} T$;

(2) 类关联关系.如果 $R=T_1.i,R=T_2.j$,则称类型 T_1 和类型 T_2 通过类型 R 关联,记为 $T_1 \xleftarrow{R} T_2$;

(3) 对象关联关系.如果 $a:R$ 且 $T_1 \xleftarrow{R} T_2$,则称类型 T_1 和类型 T_2 之间关于对象 a 关联,记为 $T_1 \xleftarrow{a:R} T_2$;

(4) 枚举关联关系.如果 B 为枚举类型,即 $v_1:B,\ldots,v_n:B$,并且 $B=T_1.i,T_2=\{B\}$,则称类型 T_1 和类型 T_2 满足枚

举关联关系,记为 $T_1 \xrightarrow{\{v_1,...,v_n\}/\langle\langle enum\rangle\rangle} T_2$;

(5) 值域关联关系.如果 *range*_exp:B,并且 $B=T_1.i,T_2=\{B\}$,则称类型 T_1 和类型 T_2 满足值域关联关系,记为 $T_1 \xrightarrow{range\_exp:B/\langle\langle range\rangle\rangle} T_2$.其中,值域表达式 *range*_exp::=$t$,$t$ 是第 2.3 节中定义的项.

对上述 5 种关系,有如下关系类的类型规则:

- 关系类规则

$$\frac{T=T_1\times...\times T_i\times...\times T_n,1\leqslant i\leqslant n}{T_i \xrightarrow{\diamond} T \text{或} T.i \xrightarrow{\diamond} T} \quad \text{(TR8)}$$

$$\frac{R=T_1.i,R=T_2.j}{T_1 \xleftrightarrow{R} T_2} \quad \text{(TR9)}$$

$$\frac{a:R,R=T_1.i,R=T_2.j}{T_1 \xleftrightarrow{a} T_2}\left(\text{或}\frac{a:R,T_1 \xleftrightarrow{R} T_2}{T_1 \xleftrightarrow{a} T_2}\right) \quad \text{(TR10)}$$

$$\frac{v_1:B,...,v_n:B,B=T_1.i,T_2=\{B\}}{T_1 \xrightarrow{\{v_1,...,v_n\}/\langle\langle enum\rangle\rangle} T_2} \quad \text{(TR11)}$$

$$\frac{t=range\_exp(t_1,...,t_n):B,B=T_1.i,T_2=\{B\}}{T_1 \xrightarrow{range\_exp/\langle\langle range\rangle\rangle} T_2} \quad \text{(TR12)}$$

2.6 项语义

DDML 语言的项语义定义如下.

(1) 设$\Gamma\vdash x:T$,则 $y\in\|T\|$,即,变量符号 x 的语义是$\|T\|$中的变量 y;

(2) 设$\Gamma\vdash c:T$,则 $u\in\|T\|$,即,常量符号 c 的语义是$\|T\|$中的某个常量 u;

(3) 设$\{l=a\}:\{l:T\}$,则$\{l=u\}\in\|\{l:T\}\|,u\in\|T\|$;

(4) 设$\Gamma\vdash t_1:T_1,...,t_n:T_n$,则$(v_1,...,v_n)\in\|T_1\|\times...\times\|T_n\|$,其中,$v_1\in\|T_1\|,...,v_n\in\|T_n\|$,“×”表示集合的笛卡尔积;

(5) 设$\Gamma\vdash t_1:T_1,...,t_n:T_n$,则$\|f(t_1,...,t_n)\|=\|f\|(v_1,...,v_n)$,其中,$v_1\in\|T_1\|,...,v_n\in\|T_n\|$,$\|f\|$是$\|T_1\|\times...\times\|T_n\|$上的映射,“×”表示集合的笛卡尔积;

(6) $\|t(u)\|=\|t\|(\|u\|)$;

(7) $\|t \text{ as } T\|=v,v\in\|T\|$;

(8) $\|\text{if } t_1 \text{ then } t_2 \text{ else } t_3\|=\text{if } \|t_1\| \text{ then } \|t_2\| \text{ else } \|t_3\|$.

2.7 基于类型理论的领域数据建模方法

为领域数据建模和算法描述方便,本文对 DDML 中的类型细分为原子类型、基本类型和复合类型,依次对应于行业数据规范中的原子类型、数据元和数据元目录.可详细定义如下:

定义 1(原子类型). 原子类型是应用给定的有限多个类型.

例如,在应用中可以定义 *string*,*integer*,*float*,*date*,*datetime*,*boolean*,*binary*,*Digit*,*Alphabet* 和 *Nat* 为原子类型.

定义 2(数据元). (1) 数据元可以是由原子类型加约束(类型规则 TR3),并且加标签(类型规则 TR1)构成的类型,其中约束可以为空;(2) 数据元可以是枚举类型;(3) 数据元可以是值域表达式类型.

定义 3(数据元目录). 数据元目录是由基本类型聚合而成(类型规则 TR4),或者由某个枚举类型或值域表达式构造而成(类型规则 TR5 或 TR6,TR7)的类型.

后面行文中,一律采用如上应用中的术语原子类型、数据元和数据元目录.

下面给出领域数据建模方法的建模框架.

- $\mathcal{K}_1=\{A_1,...,A_m\}$;
- $\mathcal{K}_2=\{B_1,...,B_n|B_j=\{l:A_i \text{ with } \langle constraint\rangle\},1\leqslant j\leqslant n,1\leqslant i\leqslant m\}$;
- $\mathcal{K}_3=\{T_1,...,T_x|T_j=B_{j1}\times...\times B_{jk}$ or $T_j=\{P_k\}$ or $T_j=\{E_k\},1\leqslant j\leqslant x,1\leqslant k\leqslant n\}$,其中,$\{P_k\}$由类型规则(TR5)定义,$\{E_k\}$由类型规则(TR6)和(TR7)定义.

$\mathcal{K}_1$称为原子类型层,$\mathcal{K}_2$称为数据元层,$\mathcal{K}_3$称为数据元目录层.$\mathcal{K}_1$,$\mathcal{K}_2$,$\mathcal{K}_3$的构造称为领域数据建模框架,$\mathcal{K}=\mathcal{K}_1\cup\mathcal{K}_2\cup\mathcal{K}_3$称为领域数据模型.如图 1 所示.

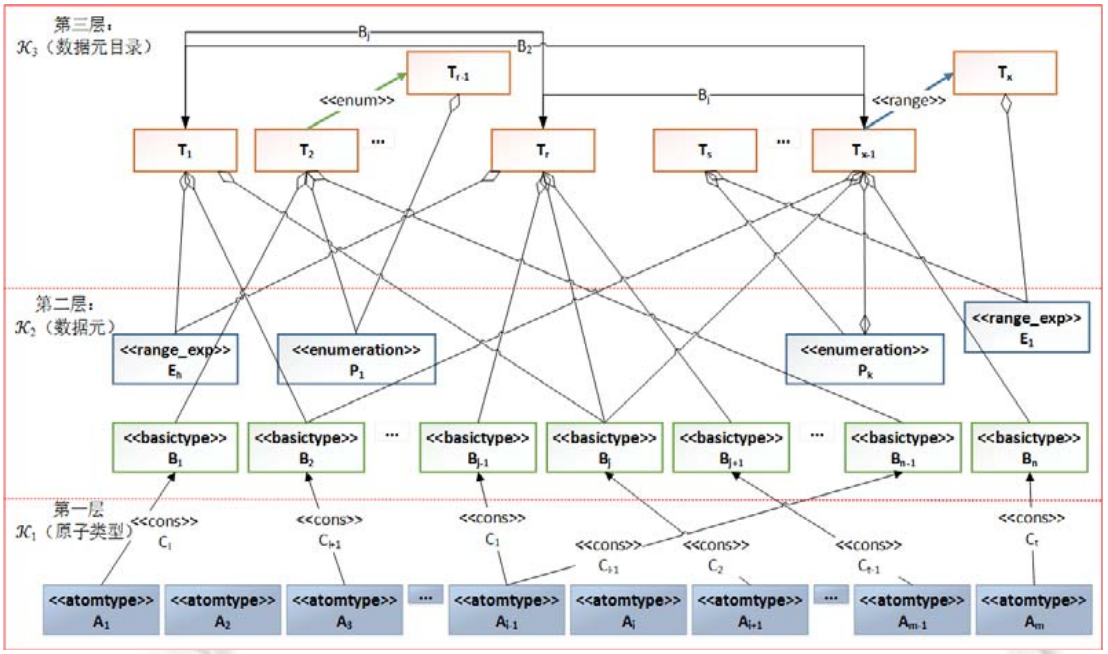


Fig.1 Domain data modelling framework

图 1 领域数据建模框架

图 1 给出的 DDMM 方法包括 4 个部分,分别为:1) 原子类型建模;2) 数据元建模;3) 数据元目录建模;4) 确定数据元目录之间的关系.其中:第一层表示数据模型中的所有原子类型$A_1,\ldots,A_m$,$m\geqslant 1$;第二层表示数据模型中的所有数据元,$B_1,\ldots,B_n$,$n\geqslant 1$(基本类型),$P_1,\ldots,P_s$,$s\geqslant 0$(枚举类型)以及 $E_1,\ldots,E_h$,$h\geqslant 0$(值域表达式类型);第三层表示模型中的所有数据元目录 $T_1,\ldots,T_x$,$x\geqslant 1$(复合类型)及其他们之间的关系,数据元目录相当于类型理论中的记录[1],它们之间的关系包括类型关联、枚举关联以及值域关联.模型中还包括两类层间关系,分别是第三层与第二层之间的聚合关联和第二层与第一层之间的约束关联,它们都是构造数据元目录的基础.而数据元目录之间的关系决定所建数据模型是否满足业务需求.

定义 4(数据元目录序列). 设 $T_k(k=1,\ldots,n)$是数据元目录,则 $T_1;\ldots;T_n$ 称为一个数据元目录序列,并且存在 T_i 和 $T_j(1\leqslant i,j\leqslant n)$满足如下类型关系之一.

- 类关联关系: $T_i \xleftrightarrow{R} T_j$;
- 对象关联关系: $T_i \xleftrightarrow{a:R} T_j$;
- 枚举关联关系: $T_i \xrightarrow{\{v_1,\ldots,v_n\}/\langle\langle enum\rangle\rangle} T_j$;
- 值域关联关系: $T_i \xrightarrow{range\_\exp:E/\langle\langle range\rangle\rangle} T_j$.

定义 5. 设$\{R_1,\ldots,R_k\}$是如上定义的数据元目录序列 $T_1;\ldots;T_n$ 对应的类型关系的集合,则 $T_1;\ldots;T_n$ 正确当且仅当 $R_1\wedge\ldots\wedge R_k$ 成立.

2.8 类型检查算法

类型检查算法的目的是判断$\Gamma\vdash t:T$是否成立,即:对任意给定的一个项t,判定其类型是否为T.为此,首先为每一条类型规则 TR1~TR7 定义其同名类型检查算法[19];然后,分别定义原子类型及约束检查算法

(CheckAtomicNConstraints)、数据元检查算法(CheckBasicType)、数据元目录检查算法(CheckComposedType);最后,利用上述 3 个算法实现对任意一个项 t 判定其类型是否为 T 的检查算法(CheckType).为便于描述算法的时间和空间复杂度,将模型中的原子类型、数据元、数据元目录、约束、标签以及类型规则的个数分别记为 C,N,M,H,K 和 P,这些参数刻画了领域问题规模.每一条类型规则 TRi 同名类型检查算法的时间和空间复杂度在文献[19]中给出.下面给出对于任意给定的项 t,其类型检查算法 CheckType 及时间和空间复杂度说明.其具体流程见算法 1.

算法 1. CheckType.

Input: t;

Output: T.

1. **begin**
2. *ResultList=null*; //初始化结果集
3. *ResultList=CheckComposedType*(t); //判断输入项是否为数据元目录
4. **if** (*ResultList*!=*null*) 输出 *ResultList* 中的 t:T **else** 继续执行步骤 5
5. *ResultList=CheckBasicType*(t); //判断输入项是否为数据元
6. **if** (*ResultList*!=*null*) 输出 *ResultList* 中的 t:T **else** 继续执行步骤 7
7. *ResultList=CheckAtomicNConstraints*(t);//判断输入项是否为原子类型或某个原子类型加约束的类型
8. **if** (*ResultList* != *null*) 输出 *ResultList* 中的 t:T **else** 继续执行步骤 9
9. 输出受阻
10. **end**

根据算法 1 中的逻辑,该算法调用了其他 3 个类型检查子算法.假设其所调用的 3 个子算法的时间复杂度分别为 O_{T1},O_{T2},O_{T3},则算法 1 的时间复杂度为 O_{T1},O_{T2},O_{T3} 中的最高者.类似地,假设其所调用的 3 个子算法的空间复杂度分别为 O_{S1},O_{S2},O_{S3},则算法 1 的空间复杂度为 O_{S1},O_{S2},O_{S3} 中的最高者.即,算法 1 的时间和空间复杂度相当于该算法所调用的 3 个子算法时间和空间复杂度中的的最大值.下面分别给出 3 个子算法的定义及其时间和空间复杂度.

(1) 子算法 1

CheckAtomicNConstraints 为原子类型及约束检查算法,CheckAtomicNConstraints 算法通过判定输入项所属的原子类型 A_i 及所有满足的约束 $C_j(j\geq 0)$(约束可以为空),给出该输入项所属的类型 A_i with $\langle C_j\rangle(i\geq 1,j\geq 0)$,约束可以为空.该算法定义如下.

算法 2. CheckAtomicNConstraints.

Input: t;

Output: T.

1. **begin**
2. *ResultList=null*; //初始化结果集
3. *ResultList=TR2*(t); //判断输入项是否为原子类型或原子类型加约束的类型
4. **if** (*ResultList*!=*null*) 输出 *ResultList* 中的 t:T **else** 继续执行步骤 5
5. 输出 受阻
6. **end**

根据上述算法的逻辑和文献[19]中 TR2 检测算法的定义,其时间复杂度 O_{T1} 为 $O(C\times H)$,空间复杂度 O_{S1} 为 $O(1)$.

(2) 子算法 2

CheckBasicType 为数据元检查算法,该算法通过判定输入项是否为带标签的数据元,处理其标签后面的项 t'的类型 B',并通过判断$\{l{=}t'\}:\{l{:}B'\}$中的$\{l{:}B'\}$是否存在于所有数据元集合中:如果存在,则认为输入项的类型为

{l:B'}所定义的数据元 B_k(k≥1),此外,数据元检查中还可以检查输入项是否为一个枚举类型值或一个值域表达式的项.该算法定义如下.

算法 3. CheckBasicType.

Input: t;

Output: T.

1. **begin**
2. *ResultList*=*null*; //初始化结果集
3. *aList*= *TR*3(t); //判断输入项是否为带标签的数据元
4. **if** (*aList*!= *null*)
   for *aList* 中的每个 t':{l:T}:
   if B={l:T}∈数据元集合输出 t:B;
   else 输出受阻;
5. **else** 继续执行步骤 6
6. 遍历枚举类型集合中的每一个 P,若 t:P,输出 t:P;否则,继续执行步骤 8
7. *bList*=*TR*6(t); //判断输入项是否为值域表达式类型
8. **if** (*bList*!=*null*) 输出 *bList* 中的 t:E, **else** 输出受阻
9. **end**

根据上述算法的逻辑和文献[19]中 TR3,TR6 检测算法的定义,其时间复杂度 O_{T2} 为 $O(C\times H\times K\times N)$,$O(N\times n)$,$O(C\times H\times K\times N\times n)$中的最高者,空间复杂度 O_{S2} 为 $O(n)$,其中,n 为构成输入项 t 的所有分项 t_i 的个数.即,n 是输入相关变量.

(3) 子算法 3

CheckComposedType 为数据元目录检查算法,该算法通过对圆括号中的每一个以符号“,”间隔的项进行判定(基本类型、枚举类型或值域表达式类型),给出这些类型所构造的数据元目录.该算法定义如下.

算法 4. CheckComposedType.

Input: t;

Output: T.

1. **begin**
2. *ResultList*=*null*; //初始化结果集
3. *ResultList*=*TR*4(t); //判断输入项是否为数据元目录
4. **if** (*ResultList*!=*null*) 输出 *ResultList* 中的 t:T **else** 继续执行步骤 5
5. *ResultList*=*TR*5(t); //判断输入项是否为枚举类型构造的数据元目录
6. **if** (*ResultList*!=*null*) 输出 *ResultList* 中的 t:T **else** 继续执行步骤 7
7. *ResultList*=*TR*7(t); //判断输入项是否为值域表达式类型构造的数据元目录
8. **if** (*ResultList*!=*null*) 输出 *ResultList* 中的 t:T **else** 输出受阻
9. **end**

根据上述算法的逻辑和文献[19]中 TR4,TR5,TR7 检测算法的定义,其时间复杂度 O_{T3} 为 $O(C\times H\times K\times n)$,$O(C\times H\times K\times N\times n)$,$O(N\times n)$,$O(C\times H\times K\times N^2\times n)$中的最高者,空间复杂度 O_{S3} 为 $O(n)$,$O(2n)$,$O(1)$中的最高者,其中,n 为构成输入项 t 的所有分项 t_i 的个数.

上述 CheckType 算法主要用于给定一个项 t,判定其所属的类型为 T.该算法的时间复杂度相当于 $O(C\times H\times K\times N^2\times n)$,空间复杂度相当于 $O(2n)$,其中,n 的含义同上.此外,数据元目录之间的关系也是检查的内容,即:给定两个数据元目录 T_1 和 T_2,判定它们之间存在什么类型关系(或给定 T_1,T_2 和一个类型关系 R 判定该关系是否满足).本文通过算法 CheckRelation 实现这一功能,该算法在所有关系类类型规则检查算法[19]的基础上定义.为便于描

述其时间和空间复杂度,模型中的数据元、数据元目录个数仍记为 N,M,数据元目录之间关系的类型规则个数记为 L.研究报告[19]中标明模型中关系类类型规则 TR8~TR13 同名检查算法的时间和空间复杂度.CheckRelation 算法具体流程如下.

算法 5. CheckRelation.

Input: T_1,T_2,o;

Output: R.

1. **begin**
2. *result*=*false*; //初始化结果,假定为不成立
3. R=*null* //可能的关系
4. **if** ((T_1,T_2,R)形如 TR9 的分母); //若符合类型关联规则的分母
   a) R=类型关联;*result*=*TR*9(T_1,T_2,R)
5. **else if** ((T_1,T_2,o)形如 TR10 的分母); //若符合对象关联规则的分母
   a) R=对象关联;*result*=*TR*10(T_1,T_2,o)
6. **else if** ((T_1,T_2,o)形如 TR11 的分母); //若符合枚举关联规则的分母
   a) R=枚举关联;*result*=*TR*11(T_1,T_2,o)
7. **else if** ((T_1,T_2,o)形如 TR12 的分母); //若符合值域类型关联规则的分母
   a) R=值域类型关联;*result*=*TR*12(T_1,T_2,o)
8. **if** (*result*==true) 输出 R //存在关系 R
9. **else** 输出 R=*null* //数据元目录无关联
10. **end**

根据上述算法的逻辑和文献[19]中 TR9~TR12 检测算法的定义,以及项的类型判定算法的效率分析,其时间复杂度相当于 $O(N\times 2M\times N^2\times L+C\times H\times K\times N^2\times n)$,空间复杂度为 $O(2N+L)$,其中,N 的含义同上,表示数据元个数.

最后,本文给出数据元目录序列正确性的判定算法,该算法通过给定的数据元目录序列及其应满足的类型关系集合,判定类型关系集合中的每个类型关系 R_i 是否满足.其具体流程如下.

算法 6. CheckCorrectness.

Input: $(T_1,...,T_m)_{\{R_1,...,R_k\}}$;

Output: boolean.

1. **begin**
2. *correctFlag*=false; //初始化结果,假定为不正确
3. 遍历$\{R_1,\ldots,R_k\}$,对每一个 R_i
4. 获取 R_i 中的 T_1,T_2,o //此处 o 为对象关联关系中的对象
5. **if** ($\{T_1;T_2\}\subseteq\{T_1;\ldots;T_m\}$) //类型关系相关的数据元目录存在
   a) R=*CheckRelation*(T_1,T_2,o)
   b) **if** (R==R_i)*correctFlag*=true
   c) **else** *correctFlag*=false
6. **else** *correctFlag*=false
7. **if** (*correctFlag*==true) 输出正确
8. **else** 输出不正确
9. **end**

根据上述算法的逻辑和 CheckRelation 算法的效率分析,其时间复杂度为 $O((N\times 2M\times N^2\times L+C\times H\times K\times N^2\times n)\times m\times k)$,空间复杂度为 $O(2N+L+m+k)$,其中,m,k 和 n 分别表示数据元目录序列长度、应满足关系集合长度以及构成算法中项 o 的所有成员个数,即,m,k 和 n 是输入相关变量.

容易看出:数据元目录之间关系和数据元目录序列正确性的判定问题最终都可以归结为判定$\Gamma\vdash t:T$是否成立的问题上,并且上述算法总是能够终止.此外,算法的时间复杂度为三次多项式复杂,空间复杂度为一次多项式复杂.DDML 语言、DDMM 方法及其类型检查算法合起来构成 DDMS(domain data modelling system)系统.关于 DDMS 系统的可判定性证明,即,其类型检查算法(CheckType,CheckRelation 和 CheckCorrectness 算法)的可靠性、完备性以及算法的可终止性的详细证明,见文献[1].

本文在上述理论与算法的基础上,设计并实现了 DDMS 原型系统工具,其总体构造如图 2 所示.

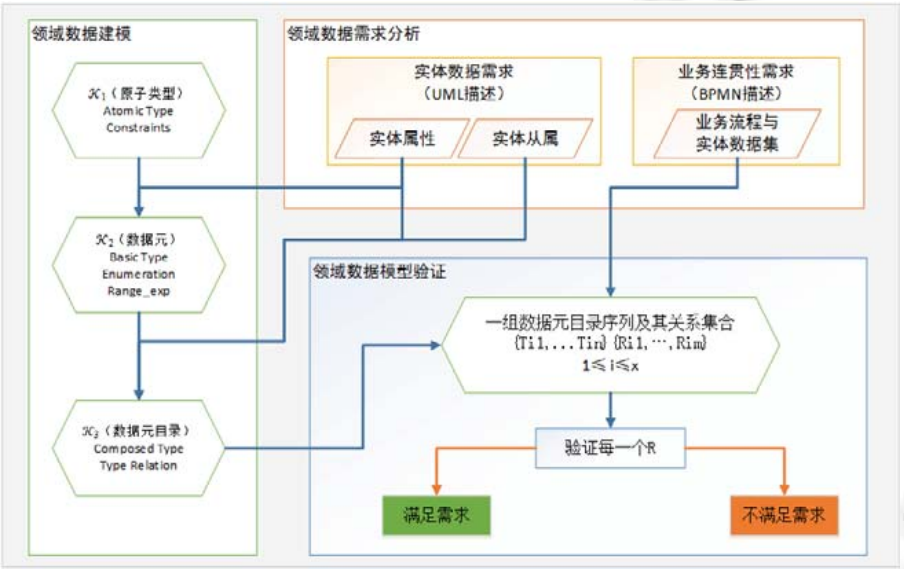


Fig.2 Concept diagram of DDMS prototype

图 2 DDMS 原型系统框架

DDMS 分为 3 个部分:领域数据需求分析、领域数据建模以及领域数据模型验证.该系统逐层创建领域各层的数据类型,同时,将需求中需要验证的性质作为验证目标输入系统.在验证过程中,系统首先对目标性质的每个逻辑单元进行判定,并最终计算出表示目标性质的命题是否成立,从而判断所创建的领域数据模型是否满足需求.

DDML 语言及 DDMM 方法可以为一般行业制定其信息系统基本数据规范并且进行验证.通过对实际行业需求和数据特点的分析,逐层生成数据模型和类型规则,形成领域数据模型,并验证这一数据模型的正确性.

3 殡葬领域数据建模案例

作为一个案例研究,本文采用 DDMS 原型系统工具构建殡葬领域数据模型,并验证该模型.

3.1 殡葬领域数据需求分析

本案例的需求为制定民政部殡葬管理与服务信息系统的基本数据规范(以下称简称数据规范),并确保其满足业务需求.该数据规范旨在为殡葬机构对其业务数据进行实时动态查询、维护、汇总和推送,为民政主管部门实时动态了解所辖地区殡葬业务运行情况、动态统计分析等提供基本数据支持.殡葬管理与服务信息系统主要包括殡葬业务管理和殡葬业务服务,其中,殡葬业务服务系统主要提供殡仪业务、安放/安葬业务、祭祀/祭扫业务等服务.数据规范是殡葬管理与服务信息系统建设的前提和基础.

数据规范的编制内容,源于本行业领域专家的知识和实际业务中的数据需求.数据规范不仅应覆盖一般实体数据,还要体现不同实体数据之间的关系.一般实体数据是指殡葬管理与服务中涉及到的实体的信息,如殡葬

机构、员工、殡葬设施/设备/用品以及殡葬业务相关信息.而实体数据之间的关系则是指实体之间是否从属、是否相关等信息.此外,对于一个整体业务服务,数据规范还应满足业务节点中的数据前后一致.

根据以上要求,编制数据规范的需求分为两部分:第 1 部分为实体数据需求;第 2 部分为业务连贯性需求.本文对原始需求采用表格的方式记录(见附录 1,完整需求见文献[19]),通过需求分析,进一步刻画为需求图.

3.1.1 实体数据需求

实体数据主要刻画实体属性和实体从属.实体属性指每个实体应具备的基本属性,用于描述一个实体由哪些元素构成.此类需求可分析为一个数据元目录由哪些数据元聚合而成.实体从属用于描述一个实体从属于哪些实体.例如,某个实体 *B* 从属于 *A*,此类需求可分析为数据元目录 *A* 与数据元目录 *B* 之间的类型关联,其中,所关联的数据元为某一个实体的标识,例如:员工从属于机构,则员工信息中有机构标识的属性;机构有设施,则设施信息中有机构标识的属性.

本文采用图形化设计工具(如 UML)对此进行描述.鉴于该部分的内容较多,限于篇幅,本文以图 3 为例展示机构信息、年检信息以及殡仪业务信息等数据元目录的构成及其关系,如图 3 所示.类似地,以图 4 为例展示数据元目录之间关系,如图 4 所示.更多详细需求,见文献[19].

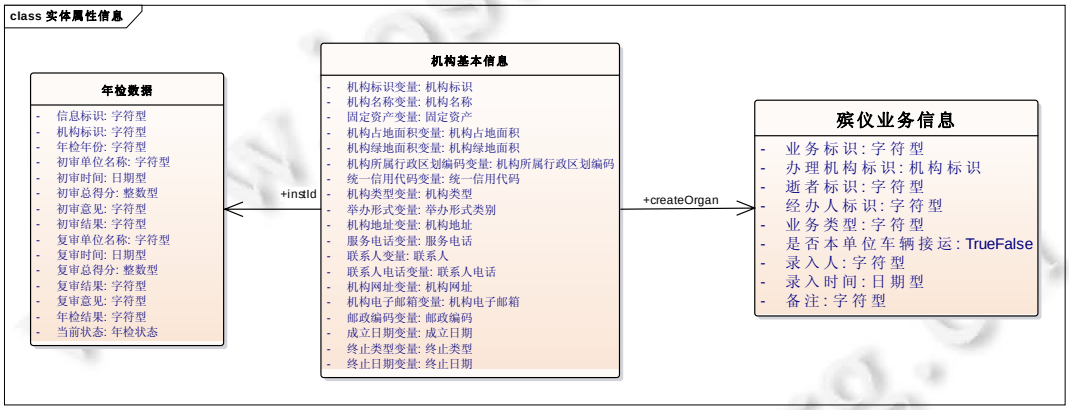


Fig.3 Requirement of data atrributes

图 3 实体属性需求

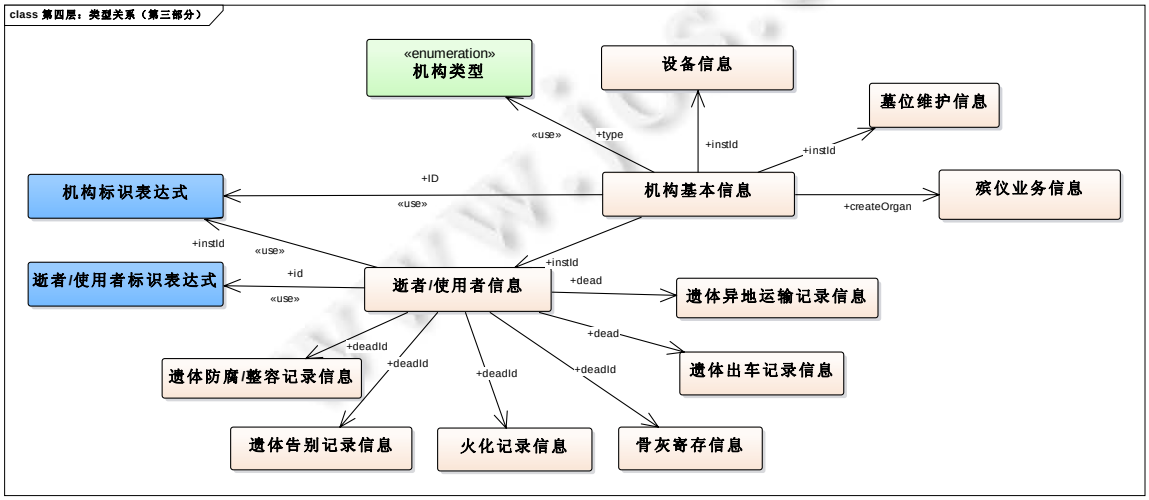


Fig.4 Requirement of data relations

图 4 实体从属需求

3.1.2 业务连贯性需求

根据本应用背景,殡葬管理与服务信息系统的业务包括 3 类:一是殡仪服务;二是安放/安葬服务;三是祭祀/祭扫服务.3 类业务均有各自的业务流程,每个业务流程通常由多个分支组成,并且每个分支上存在多个节点.要求这 3 个业务流程均能满足其所有分支上的节点所需数据存在,并且要求前后节点数据之间能够关联.例如:在殡仪业务流程中,办理遗体接运、保存、守灵、防腐、整容以及火化、取灰等一系列节点中,前后节点的机构、设备、设施或逝者信息应统一.

因此,业务连贯性需求主要描述每类业务流程的各个分支需要哪些数据以及前后节点的数据之间是否有联系,即,每类业务的各个分支中应满足一系列数据元目录序列以及数据元目录之间的关系.本文采用图形化建模工具,如 BPMN(business process modelling notation),详细描述.其中,以殡仪业务为例,其业务流程中所需要的数据如图 5 所示.

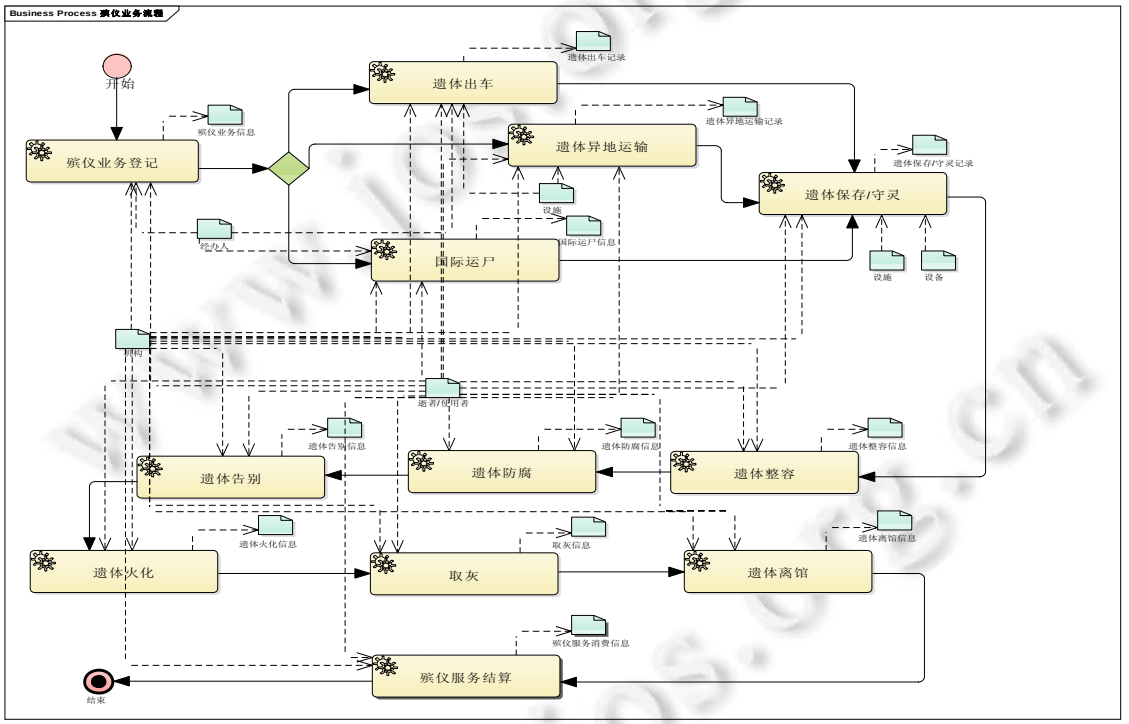


Fig.5 Data requirements of business process

图 5 业务连贯性需求

图 5 中:虚线表示该服务节点上所需输入的数据和所要输出的数据信息;实线表示业务流程的顺序.图中的开始节点到结束节点形成一个业务分支,每个业务分支由多个服务节点组成.以图 5 为例,殡仪业务流程包括 10 个服务节点,其中,“遗体接运”服务又可以划分为“遗体出车”“国际运尸”和“遗体异地运输”这 3 个分支,因此,殡仪业务流程包括从开始节点到结束节点的 3 条业务分支.若依次记录服务节点为 $S_1,S_2,S_3,S_4,S_5,S_6,S_7,S_8,S_9,S_{10}$ (其中, $S_2 = S_2^1 \mid S_2^2 \mid S_2^3$ 表示三者选一),则殡仪业务连贯性可以视为每个 S_i 需要的输入/输出数据是否定义以及它们之间关系是否得到满足.图 5 中,每个业务分支的不同服务节点都以同一个“机构”数据与“逝者/使用者”数据作为输入,表示该分支中“机构标识”与“逝者/使用者标识”前后一致.类似地,其他两个服务的业务流程的图形化展示见文献[19].

将 UML 和 BPMN 刻画的需求图进行导出,可以转化为程序可处理的 XML 文件,通过解析这些 XML 文件,可以得到响应的实体数据需求点和业务连贯性需求点.关于如何导出以及如何解析的方法及理论,限于篇幅,本

文不再赘述.

3.2 殡葬领域数据建模

为满足上述需求,DDMS 原型系统采用 DDML 语言及 DDMM 方法,对殡葬领域数据进行建模.建模包括$\mathcal{K}_1$原子类型及约束,$\mathcal{K}_2$数据元、枚举类型、值域表达式类型以及$\mathcal{K}_3$数据元目录及其关系.

3.2.1 原子类型及约束

原子类型是殡葬行业数据类型的基础,它包括,*string*,*integer*,*float*,*date*,*datetime*,*boolean*,*binary*,*Digit*,*Alphabet* 和 *Nat* 等 10 种类型,分别编号为 A_1~A_{10}.它们构成数据模型中$\mathcal{K}_1$层,其类型总数为 10 个.为方便在实际规范中引用,在应用中采用以下更为直观的约束表达式形式.

由第 2.1 节中关于类型的定义,约束表达式⟨*constraint*⟩用于限定某一类对象,其形式定义为

constraint::=*number*|…*number*|*number*$_1$…*number*$_2$|*n number*|*n*…*number*|*an number*|,

其中,*number*:*Nat*,*number*$_1$:*Nat*,*number*$_2$:*Nat*,*a* as *Alphabeta*(||*Alphabet*||={*a*,…,*z*,*A*,…,*Z*}),*n* as *Digit*(||*Digit*||={0,1,…,9}),*an* as(*Alphabeta*+*Digit*)*.as 表示类型归属,归属也是一种类型[1].容易看出,每个约束表达式相当于 DDML 中约束的一个实例.表 1 给出了应用中的约束形式及其在 DDML 中对应的约束表达式.

Table 1 List of constraints

表 1 约束列表

约束形式	说明	DDML 约束表达式
number	表示对象 *t* 的长度约束(或符号个数)为固定值 *number*	*length*(*t*)=*number*
…*number*	表示对象 *t* 的长度约束(或符号个数)为大于等于 0,小于等于 *number*	0≤*length*(*t*)≤*number*
number$_1$…*number*$_2$	表示对象 *t* 的长度(或符号个数)约束为大于等于 *number*$_1$,小于等于 *number*$_2$	*Number*$_1$≤*length*(*t*)≤*number*$_2$
nnumber	表示对象 *t* 为数字符号串,且长度约束(或符号个数)为固定值 *number*	*t*∈{0,1,…,9}*,*length*(*t*)=*number*
n…*number*	表示对象 *t* 为数字符号串,且长度约束(或符号个数)为大于等于 0,小于等于 *number*	*t*∈{0,1,…,9}*,0≤*length*(*t*)≤*number*
an number	表示对象 *t* 为字母和数字符号串,且长度约束(或符号个数)为固定值 *number*	*t*∈({*a*,…,*z*,*A*,…,*Z*}∪{0,1,…,9})*,0≤*length*(*t*)≤*number*

3.2.2 数据元

由第 2.7 节中定义 2,数据元由原子类型加上标签和约束表达式构造而成,其中,约束表达式可以为空.

应用中的数据元集合记为{B_1,…,B_n},其中,B_i=l_i:A_j with ⟨*constraint*$_i$⟩(0≤*i*≤*n*).为满足附录 1 和文献[19]中记述的需求,以下举例给出本文创建和表述数据元的方式.

例 1:*string* with ⟨*an*16⟩表示长度为 16 的英文字母加数字符号串类型.应用中,机构标识 B_1 满足该约束,其标识为“*instId*”,因此有 B_1=*instId*:A_1 with ⟨*an*16⟩.

例 2:表 2 为本文定义数据元的方式,其中,表头的“数据元编号”表示第 *i* 个数据元 B_i;第 1 行中的标签“*instCode*”和约束⟨*an*18⟩表示数据元 B_{12}(“统一信用代码”)是 18 位字母与数字组合的字符串类型,因此,B_{12}=A_1 with ⟨*an*18⟩;第 2 行中的标签“*useLimit*”和约束⟨*n*…3⟩,表示数据元 B_{48}(“使用年限”)是 3 位数字以内的整数类型,因此有 B_{48}=A_2 with ⟨*n*…3⟩.采用这一方式,DDMS 原型系统生成了《原子类型基本类型定义》[19],它给出了殡葬行业数据规范需要的所有数据元的定义,它们构成数据模型中的$\mathcal{K}_2$层,其类型总数为 549 个.

Table 2 List of data element

表 2 数据元列表

数据元编号	中文名称	标签	原子类型	约束
B_{12}	统一信用代码	instCode	*string*	*an*18
B_{48}	使用年限	useLimit	*integer*	*n*…3

3.2.3 数据元目录

由第 2.7 节中定义 3,数据元目录是由数据元聚合而成的类型,或者由某个枚举类型或值域表达式构造而成的类型.

应用中的数据元目录集合记为$\{T_1,\ldots,T_x\}$,其中,$T_j=B_{j1}\times\ldots\times B_{jk}(1\leqslant j\leqslant x,1\leqslant k\leqslant n).B_{jk}$ 可以是数据元,也可以是枚举类型或值域表达式类型.

设有如下原子类型或数据元的项:

$$a,b,y,m,d,x:A_8,b_{11}:B_{11},b_{12}:B_{12},b_{13}:B_{13},b_{274}:B_{274},b_{283}:B_{283},\ b_{308}:B_{308},$$

则根据第 2.5 节中值域表达式的定义,可以构造本应用中的值域表达式,其定义如下:

$range_$exp::=$aaaaaaaaaaaa|xxxx\text{-}aaaaaaaa\text{-}bbb|b_{11}\_b_{13}\_xxxxxx|$

‘S’$\_b_{11}\_b_{12}\_xxxxxx|$‘C’$\_b_{11}\_b_{12}\_xxxxxxxx|$‘E’$\_b_{11}\_yyy\_xxxxxx|$

‘IM’$\_b_{11}\_b_{12}\_b_{274}\_xxxxxx|$‘$EQ$’$\_b_{11}\_b_{12}\_b_{283}\_xxxxxx|$

‘Y’$\_b_{11}\_b_{12}\_b_{308}\_xxxxxx|$‘$F$’$\_b_{11}\_b_{12}\_yyymmdd\_xxxxxx|$

‘B’$\_b_{11}\_b_{12}\_yyymmdd\_xxxxxx|$‘$T$’$\_b_{11}\_b_{12}\_xxxxxxxx|$

‘D’$\_b_{11}\_b_{12}\_yyymmdd\_xxxxxx$.

若对每个表达式分别记为 $E_1,E_2,\ldots,E_{13}$,则本应用中值域表达式类型有 13 个,其对应的数据元,见表 3,其中,第 3 列“构成元素”表示该表达式是将这些构成元素以字符“_”(或“-”)分隔并串连而成的.

Table 3 List of range expressions

表 3 值域表达式列表

数据元	中文名称	值域表达式	构成元素
B_3	地理编码	E_1	标准 12 位地理编码
B_{18}	联系电话	E_2	4 位区号,8 位座机号和 3 位分机号
B_1	机构标识	E_3	6 位行政编码,2 位机构类型和 6 位顺序号
B_{50}	人员编号	E_4	‘S’,6 位地政编码,18 位统一信用代码和 6 位顺序号
B_{175}	信息标识	E_5	‘C’,6 位行政编码,18 位统一信用代码和 6 位顺序号
B_{253}	突发事件标识	E_6	‘E’,6 位地区行政编码,4 位年和 6 位顺序号
B_{272}	设施标识	E_7	‘IM’,6 位行政编码,18 位统一信用代码,2 位设施类型和 6 位顺序号
B_{281}	设备标识	E_8	‘EQ’,6 位行政编码,18 位统一信用代码,3 位设备类型和 6 位顺序号
B_{306}	用品标识	E_9	‘Y’,6 位行政编码,18 位统一信用代码,2 位用品类型和 6 位顺序号
B_{326}	经办人标识	E_{10}	‘F’,6 位行政编码,18 位统一信用代码,8 位年月日和 6 位顺序号
B_{406}	消费记录标识	E_{11}	‘B’,6 位行政编码,18 位统一信用代码,8 位年月日和 6 位顺序号
B_{523}	墓位标识	E_{12}	‘T’,6 位行政编码,18 位统一信用代码和 8 位顺序号
B_{524}	逝者/使用者标识	E_{13}	‘D’,6 位行政编码,18 位统一信用代码 8 位年月日和 6 位顺序号

在 DDMS 原型系统生成的原子类型基本类型定义[19]中,给出了殡葬行业数据规范需要的所有枚举类型及值域表达式类型的定义,并在值域中列出了每个类型的项可能的取值或对应的值域表达式编号.

为满足附录 1 和文献[19]中记述的需求,本文以电子表格及其宏定义程序的方式给出应用中创建的数据元目录.

例 3:表 4 中,每个数据元目录为电子表格中的一个表单,该表单有纵向表头和横向表头.其中,纵向表头中的类型名称为数据元目录的名称,类型编号为数据元目录的编号,浅灰色表头下面每行表示该数据元目录由哪些数据元构成以及这些数据元又由哪些原子类型加约束构成.将横纵信息结合,可在 TypeExp 中计算出该数据元目录的类型表达式.例如:数据元目录机构基本信息 T_2 由 20 余种数据元聚合而成,包括机构标识类 $B_1,\ldots$,机构类型类 $B_{13},\ldots$,服务范围 B_{93} 等.

采用这一方式,DDMS 原型系统生成了复合类型定义[19],它给出了殡葬行业数据规范需要的所有数据元目录,其中有 76 个由基本类型聚合而成数据元目录,106 个由枚举类型或值域表达式类型构造而成的数据元目录.它们构成数据模型中的$\mathcal{K}_3$层,其数据元目录总数为 182 个.

Table 4 Table of a data element directory

表 4 数据元目录表

2	类型名称	机构基本信息规范 InstBaseInfo						
	类型编号	T_2						
	Type Exp	$T_2=B_1\times B_7\times B_8\times B_9\times B_{10}\times B_{11}\times B_{12}\times B_{13}\times B_{14}\times B_{15}\times B_{16}\times B_{17}\times B_{18}\times B_{19}\times B_{20}\times B_{21}\times B_{22}\times B_{136}\times B_{137}\times B_{93}$						
	TR 分子	$b_1{:}B_1,b_7{:}B_7,b_8{:}B_8,b_9{:}B_9,b_{10}{:}B_{10},b_{11}{:}B_{11},b_{12}{:}B_{12},b_{13}{:}B_{13},b_{14}{:}B_{14},b_{15}{:}B_{15},b_{16}{:}B_{16},b_{17}{:}B_{17},b_{18}{:}B_{18},b_{19}{:}B_{19},b_{20}{:}B_{20},b_{21}{:}B_{21},b_{22}{:}B_{22},b_{136}{:}B_{136},b_{137}{:}B_{137},b_{93}{:}B_{93}$						
	TR 分母	$(b_1,b_7,b_8,b_9,b_{10},b_{11},b_{12},b_{13},b_{14},b_{15},b_{16},b_{17},b_{18},b_{19},b_{20},b_{21},b_{22},b_{136},b_{137},b_{93})$: $T_2,T_2=B_1\times B_7\times B_8\times B_9\times B_{10}\times B_{11}\times B_{12}\times B_{13}\times B_{14}\times B_{15}\times B_{16}\times B_{17}\times B_{18}\times B_{19}\times B_{20}\times B_{21}\times B_{22}\times B_{136}\times B_{137}\times B_{93}$						
基本类型编号	内部标识	中文名称	标签	类型	格式	值域	构成关系	格式要求
B_1	AQAT0001	机构标识	instId	*string*	*an*16	*E2*	A_1	,⟨*an*16⟩
B_7	AQAT0007	机构名称	name	*string*	…256		A_1	,⟨…256⟩
B_8	AQAT0008	固定资产	fixedAssets	*float*			A_3	,⟨*float*⟩
B_9	AQAT0009	机构占地面积	scale	*float*			A_3	,⟨*float*⟩
B_{10}	AQAT0010	机构绿地面积	greenScale	*float*			A_3	,⟨*float*⟩
B_{11}	AQAT0011	机构所属行政区划编码	areaCode	*string*	*n*12		A_1	,⟨*n*12⟩
B_{12}	AQAT0012	统一信用代码	instCode	*string*	*an*9		A_1	,⟨*an*9⟩
…	…	…	…	…	…		…	…
B_{93}	AQAT0093	服务范围	svcScope	*string*	…1024		A_1	,⟨…1024⟩

3.2.4 类型关系及类型规则

领域数据模型中的各层类型之间存在一定的关系,类型关系是指每层类型之间以及某些层内类型之间的关系.类型关系既可以采用类型表达式,也可以采用类型规则描述.若采用类型表达式,应用中的类型关系可以写为

$$R_1(T_{11},\ldots,T_{1i}),\ldots,R_k(T_{k1},\ldots,T_{kj}),$$

其中,$\{T_{11},\ldots,T_{1i}\}\subseteq\{T_1,\ldots,T_m\},\ldots,\{T_{k1},\ldots,T_{kj}\}\subseteq\{T_1,\ldots,T_x\}$.

为类型关系描述直观和算法处理方便,本文采用类型规则描述,并对其进行如下分类.

1) 在第 2 层与第 1 层之间,约束关系的类型规则;

2) 在第 3 层与第 2 层之间,聚合关系的类型规则;

3) 在第 3 层数据元目录之间的一般关联、枚举关联和值域表达式关联等关系的类型规则.

根据第 2.5 节定义的类型规则,在本应用案例中有如下类型规则.

(1) 约束关系的类型规则

数据元由原子类型加约束构成,因此,原子类型与数据元之间存在约束关联.例如,表 4 中记录了数据元机构标识(B_1)的构成关系(A_1)和格式要求(⟨*an*16⟩),说明原子类型 A_1 和数据元 B_1 之间存在约束关系,即 $A_1\xrightarrow{\langle an16\rangle}B_1$. 为便于程序处理,将类型的下标均写为正常序号,因此,表 4 中 A_1=A_1,B_1=B_1.

本应用中的约束关系的类型规则总共有 480 个.

(2) 聚合关系的类型规则

数据元目录由数据元聚合而成,因此数据元与数据元目录之间存在聚合关系.例如,表 4 中记录了数据元目录(T_2)与 20 多个数据元($B_1,\ldots,B_{22},\ldots,B_{93}$),并在“TR 分子”和“TR 分母”中给出了当前数据元目录由数据元聚合而成的类型规则,如下所示:

$$\frac{T=B_1\times\ldots\times B_{13}\times\ldots\times B_{22}\times B_{93}}{B_1\xrightarrow{\diamond}T\text{或}T.1\xrightarrow{\diamond}T},\ldots,\frac{T=B_1\times\ldots\times B_{13}\times\ldots\times B_{22}\times B_{93}}{B_{13}\xrightarrow{\diamond}\text{或}T.13\xrightarrow{\diamond}T},\ldots,$$

$$\frac{T=B_1\times\ldots\times B_{13}\times\ldots\times B_{22}\times B_{93}}{B_{22}\xrightarrow{\diamond}T\text{或}T.18\xrightarrow{\diamond}T},\ldots,\frac{T=B_1\times\ldots\times B_{13}\times\ldots\times B_{22}\times B_{93}}{B_{93}\xrightarrow{\diamond}\text{或}T.21\xrightarrow{\diamond}T}.$$

本应用中的聚合关系的类型规则有 706 个.

(3) 一般关联

一般关联指 DDML 语言的类型关联和对象关联.在本应用中,数据元目录 T_{15} 和数据元目录 T_2 的元组中都有数据元 B_1(见文献[19]),因此,它们之间存在类型关联,其类型规则如下所示:

$$\frac{B_1=T_{15}.2, B_1=T_2.1}{T_{15}\xleftrightarrow{B_1}T_2}.$$

并且,对于对象 $instId$:B_1,存在它们之间的对象关联,其类型规则如下所示:

$$\frac{instId: B_1, B_1=T_{15}.2, B_1=T_2.1}{T_{15}\xleftrightarrow{instId}T_2}.$$

本应用中的类型关联有 150 个.

(4) 枚举关联

枚举关联是指枚举类型构造的数据元目录和引用该枚举类型的数据元目录之间的关联.在本文规范中,机构设施信息的数据元目录类型为 T_{14},其第 9 个元组的类型为数据元 B_{279}(“设施状态”),它的值域通过列举枚举类型 T_{124}(“设施状态类别”)值的方式给出.因此,类型 T_{14} 和类型 T_{124} 满足⟨⟨$enum$⟩⟩关联,其类型规则如下所示:

$$\frac{"01": B_{279}, "02": B_{279}, "03": B_{279}, T_{14}.j=B_{279}, T_{115}=\{B_{279}\}}{T_{14}\xrightarrow{\{"01","02","03"\}/\langle\langle enum\rangle\rangle}T_{124}}.$$

本应用中,枚举关联有 100 个.

(5) 值域表达式关联

值域关联是指值域表达式类型构造的数据元目录和引用该值域表达式类型的数据元目录之间的关联.结合表 3 和表 4,机构基本信息的数据元目录类型 T_2 的第一元组类型为数据元 B_1(“机构标识”),其值域由值域表达式 E_2 给出.因此,数据元目录 T_2 与值域表达式构造的数据元目录 T_{171} 之间存在值域关联,其类型规则如下所示:

$$\frac{t: E_2, T_{171}=\{E_2\}, T_2.j=E_2}{T_2\xrightarrow{t/\langle\langle range\rangle\rangle}T_{171}}.$$

本应用中,值域关联有 156 个.

3.2.5 应用环境

根据 DDML 语言中环境的定义,殡葬领域数据模型中的应用环境定义如下.

约定. 将$\mathcal{K}_1$ 层的原子类型称为初始应用环境,记为Γ_1.领域数据模型中的所有类型的集合称为应用环境,记为Γ.

设Γ_2 表示数据元,Γ_3 表示数据元目录,则根据定义 8,$\Gamma=\Gamma_1\cup\Gamma_2\cup\Gamma_3$.本文所要验证的性质就是在该应用环境$\Gamma$下的性质.

3.3 殡葬领域数据模型验证

本节给出领域数据模型需要验证的性质,并采用 DDMS 原型系统对所创建模型进行验证.其主要方法是将数据模型所期望的性质 P,根据第 3.1 节中需求图的描述及其的解析,转化为判定一组数据元目录序列是否正确,并利用 DDMM 中的数据元目录序列正确性判定算法(CheckCorectness),验证 P 是否成立.

根据文献[19]中数据元目录的编号及其构成以及第 3 节中对业务流程中数据需求的分析,可以将每个业务流程中需要的数据视为该业务流程需要的数据元目录序列,将前后服务节点数据的一致性要求视为相应服务节点的数据元目录之间应满足对象关联.假定 3 种业务流程应该满足的性质分别记为 P_1,P_2 和 P_3,分别表示殡仪业务流程、安放/安葬业务流程、祭祀/祭扫业务流程所需数据元目录序列及其应满足关系集合.

为模拟业务连贯性需求,对每个业务分支中每个服务节点的数据需求进行刻画,形成表 5 的数据元目录序列及其类型关系集合.

表 5 中每一行数据元目录序列及其关系集合所表达的具体含义见表 6.

篇幅所限,表 6 的详细内容请见文献[19]中验证关系含义.

Table 5 Sequences of data element directory and its relations

表 5 数据元目录序列及其关系集合

序列编号	数据元目录序列	集合名称	类型关系	含义代码序列
SEQ1	$T_2;T_{23};T_{24};T_{51}$	R_{SEQ1}	$R_{11}=T_2\xleftrightarrow{B_1}T_{23},R_{12}=T_2\xleftrightarrow{B_1}T_{24},R_{13}=T_{51}\xleftrightarrow{B_{524}}T_{24}$	$C_1;C_2;C_3$
SEQ21	$T_2;T_{23};T_{24};$ $T_{51};T_{15};T_{52}$	R_{SEQ21}	$R_{21}=T_{52}\xleftrightarrow{b_1:B_1}T_2,R_{22}=T_{51}\xleftrightarrow{b_{524}:B_{524}}T_{52},R_{23}=T_{52}\xleftrightarrow{b_{326}:B_{326}}T_{24},$ $R_{24}=T_{52}\xleftrightarrow{b_{524}:B_{524}}T_{23},R_{25}=T_{52}\xleftrightarrow{B_{281}}T_{15}$	$C_4;C_5;$ $C_6;C_7;C_8$
SEQ22	$T_2;T_{23};T_{51};$ $T_{15};T_{53}$	R_{SEQ22}	$R_{26}=T_{53}\xleftrightarrow{b_1:B_1}T_2,R_{27}=T_{51}\xleftrightarrow{b_{524}:B_{524}}T_{53},$ $R_{28}=T_{53}\xleftrightarrow{b_{524}:B_{524}}T_{23},R_{29}=T_{53}\xleftrightarrow{B_{281}}T_{15}$	$C_9;C_{10};C_{11};$ $C_{12};C_{13}$
SEQ23	$T_2;T_{23};T_{24};$ $T_{51};T_{54}$	R_{SEQ23}	$R_{231}=T_{54}\xleftrightarrow{b_1:B_1}T_2,R_{232}=T_{51}\xleftrightarrow{b_{524}:B_{524}}T_{54},$ $R_{233}=T_{54}\xleftrightarrow{b_{326}:B_{326}}T_{24},R_{234}=T_{54}\xleftrightarrow{b_{524}:B_{524}}T_{23}$	$C_{14};C_{15};$ $C_{16};C_{17}$
SEQ3	$T_2;T_{23};T_{14};T_{15};$ $T_{52};T_{53};T_{54};T_{55}$	R_{SEQ3}	$R_{31}=T_{55}\xleftrightarrow{b_{524}:B_{524}}T_{52},R_{32}=T_{55}\xleftrightarrow{b_{524}:B_{524}}T_{53},$ $R_{33}=T_{55}\xleftrightarrow{b_{524}:B_{524}}T_{54},R_{34}=T_{55}\xleftrightarrow{B_1}T_2,$ $R_{35}=T_{55}\xleftrightarrow{B_{524}}T_{23},R_{36}=T_{55}\xleftrightarrow{B_{281}}T_{15},R_{37}=T_{55}\xleftrightarrow{B_{272}}T_{14}$	$C_{18};C_{19};$ $C_{20};C_{21};$ $C_{22};C_{23};C_{24}$
SEQ4	$T_2;T_{23};T_{55};T_{56}$	R_{SEQ4}	$R_{41}=T_2\xleftrightarrow{B_1}T_{56},R_{42}=T_{56}\xleftrightarrow{B_{524}}T_{23},R_{43}=T_{55}\xleftrightarrow{b_{524}:B_{524}}T_{56}$	$C_{25};C_{26};C_{27}$
SEQ5	$T_2;T_{23};T_{56};T_{57}$	R_{SEQ5}	$R_{51}=T_2\xleftrightarrow{B_1}T_{57},R_{52}=T_{57}\xleftrightarrow{B_{524}}T_{23},R_{53}=T_{56}\xleftrightarrow{b_{524}:B_{524}}T_{57}$	$C_{28};C_{29};C_{30}$
SEQ6	$T_2;T_{23};T_{57};T_{58}$	R_{SEQ6}	$R_{61}=T_2\xleftrightarrow{B_1}T_{58},R_{62}=T_{58}\xleftrightarrow{B_{524}}T_{23},R_{63}=T_{57}\xleftrightarrow{b_{524}:B_{524}}T_{58}$	$C_{31};C_{32};C_{33}$
SEQ7	$T_2;T_{23};T_{58};T_{59}$	R_{SEQ7}	$R_{71}=T_2\xleftrightarrow{B_1}T_{59},R_{72}=T_{59}\xleftrightarrow{B_{524}}T_{23},R_{73}=T_{58}\xleftrightarrow{b_{524}:B_{524}}T_{59}$	$C_{34};C_{35};C_{36}$
SEQ8	$T_2;T_{23};T_{59};T_{60}$	R_{SEQ8}	$R_{81}=T_2\xleftrightarrow{B_1}T_{60},R_{82}=T_{60}\xleftrightarrow{B_{524}}T_{23},R_{83}=T_{59}\xleftrightarrow{b_{524}:B_{524}}T_{60}$	$C_{37};C_{38};C_{39}$
SEQ9	$T_2;T_{23};T_{60};T_{61}$	R_{SEQ9}	$R_{91}=T_2\xleftrightarrow{B_1}T_{61},R_{92}=T_{61}\xleftrightarrow{B_{524}}T_{23},R_{93}=T_{60}\xleftrightarrow{b_{524}:B_{524}}T_{61}$	$C_{40};C_{41};C_{42}$
SEQ10	$T_2;T_{23};T_{61};T_{62}$	R_{SEQ10}	$R_{101}=T_2\xleftrightarrow{B_1}T_{62},R_{102}=T_{62}\xleftrightarrow{B_{524}}T_{23},R_{103}=T_{61}\xleftrightarrow{b_{524}:B_{524}}T_{62}$	$C_{43};C_{44};C_{45}$
SEQ11	$T_2;T_{24};T_{64}$	R_{SEQ11}	$R_{611}=T_2\xleftrightarrow{B_1}T_{24},R_{612}=T_{64}\xleftrightarrow{B_{326}}T_{24},R_{613}=T_{64}\xleftrightarrow{B_1}T_2$	$C_2;C_{48};C_{49}$
SEQ12	$T_2;T_{24};T_{64};T_{65}$	R_{SEQ12}	$R_{621}=T_2\xleftrightarrow{B_1}T_{24},R_{622}=T_{65}\xleftrightarrow{B_{326}}T_{24},R_{623}=T_{65}\xleftrightarrow{B_1}T_2$	$C_2;C_{50};C_{51}$
SEQ13	$T_2;T_{24};T_{65};T_{66}$	R_{SEQ13}	$R_{631}=T_2\xleftrightarrow{B_1}T_{24},R_{632}=T_{66}\xleftrightarrow{B_{326}}T_{24},R_{633}=T_{66}\xleftrightarrow{B_1}T_2$	$C_2;C_{52};C_{53}$
SEQ14	$T_2;T_{23};T_{66};T_{67}$	R_{SEQ14}	$R_{641}=T_2\xleftrightarrow{B_1}T_{23},R_{642}=T_{67}\xleftrightarrow{B_{524}}T_{23},R_{643}=T_{67}\xleftrightarrow{B_1}T_2$	$C_1;C_{54};C_{55}$
SEQ15	$T_2;T_{23};T_{67};T_{71}$	R_{SEQ15}	$R_{651}=T_2\xleftrightarrow{B_1}T_{23},R_{652}=T_2\xleftrightarrow{B_1}T_{71},$ $R_{653}=T_{71}\xleftrightarrow{b_{524}:B_{524}}T_{67},R_{654}=T_{71}\xleftrightarrow{B_{524}}T_{23}$	$C_1;C_{56};$ $C_{57};C_{58}$
SEQ16	$T_2;T_{23};T_{24};$ $T_{71};T_{72}$	R_{SEQ16}	$R_{661}=T_2\xleftrightarrow{B_1}T_{23},R_{662}=T_2\xleftrightarrow{B_1}T_{72},R_{663}=T_{72}\xleftrightarrow{B_{524}}T_{23},$ $R_{664}=T_{72}\xleftrightarrow{B_{326}}T_{24},R_{665}=T_{72}\xleftrightarrow{b_{524}:B_{524}}T_{71}$	$C_1;C_{59};C_{60};$ $C_{61};C_{62}$
SEQ17	$T_2;T_{23};T_{24};$ $T_{69};T_{68}$	R_{SEQ17}	$R_{671}=T_2\xleftrightarrow{B_1}T_{23},R_{672}=T_2\xleftrightarrow{B_1}T_{24},R_{673}=T_{68}\xleftrightarrow{B_{524}}T_{23},$ $R_{674}=T_{68}\xleftrightarrow{B_1}T_2,R_{675}=T_{68}\xleftrightarrow{B_{326}}T_{24},R_{676}=T_{68}\xleftrightarrow{b_{524}:B_{524}}T_{69}$	$C_1;C_2;C_{67};$ $C_{68};C_{69};C_{70}$
SEQ18	$T_2;T_{23};T_{69}$	R_{SEQ18}	$R_{681}=T_2\xleftrightarrow{B_1}T_{23},R_{682}=T_{69}\xleftrightarrow{B_{524}}T_{23},R_{683}=T_{69}\xleftrightarrow{B_1}T_2$	$C_1;C_{71};C_{72}$
SEQ19	$T_2;T_{73};T_{70}$	R_{SEQ19}	$R_{695}=T_2\xleftrightarrow{B_1}T_{23},R_{691}=T_{70}\xleftrightarrow{B_1}T_2,R_{697}=T_{70}\xleftrightarrow{B_{524}}T_{23}$	$C_1;C_{73};C_{74}$
SEQ20	$T_2;T_{23};T_{73}$	R_{SEQ20}	$R_{695}=T_2\xleftrightarrow{B_1}T_{23},R_{696}=T_{73}\xleftrightarrow{B_1}T_2,R_{697}=T_{73}\xleftrightarrow{B_{524}}T_{23}$	$C_1;C_{75};C_{76}$
SEQ31	$T_2;T_{23};T_{24};T_{74}$	R_{SEQ31}	$R_{711}=T_2\xleftrightarrow{B_1}T_{23},R_{712}=T_{74}\xleftrightarrow{B_1}T_2,$ $R_{713}=T_{74}\xleftrightarrow{B_{524}}T_{23},R_{714}=T_{74}\xleftrightarrow{B_{326}}T_{24}$	$C_1;C_{77};$ $C_{78};C_{79}$
SEQ32	$T_2;T_{23};T_{24};$ $T_{29};T_{75}$	R_{SEQ32}	$R_{721}=T_2\xleftrightarrow{B_1}T_{23},R_{722}=T_{75}\xleftrightarrow{B_1}T_2,R_{723}=T_{75}\xleftrightarrow{B_{524}}T_{23},$ $R_{724}=T_{75}\xleftrightarrow{B_{326}}T_{24},R_{725}=T_{75}\xleftrightarrow{B_{89}}T_{29}$	$C_1;C_{80};C_{81}$
SEQ33	$T_2;T_{76}$	R_{SEQ33}	$R_{731}=T_{76}\xleftrightarrow{B_1}T_2$	C_{84}

Table 6 Meaning of the relations of the data element directory sequences

表 6 数据元目录序列的关系集合具体含义

含义代码	描述
C_1	逝者信息与机构信息应能够通过机构标识关联
C_2	经办人信息与机构信息应能够通过机构标识关联
C_3	逝者信息与殡仪登记信息通过逝者标识关联
C_4	遗体接运信息应与殡仪业务登记信息通过机构标识对象关联,表示在同一机构办理
C_5	遗体接运信息应与殡仪业务登记信息通过逝者标识对象关联,表示处理的是同一位逝者
C_6	遗体接运信息应与经办人信息通过逝者标识对象关联,表示处理的是同一位逝者
…	…
C_{83}	网上祭祀祭扫信息与逝者照片信息应能够通过逝者图片标识关联
C_{84}	祭祀祭扫管理信息与机构信息应能够通过机构标识关联

结合上述表 5 和表 6,殡仪业务流程连贯性需求 P_1 可以写为如下形式:

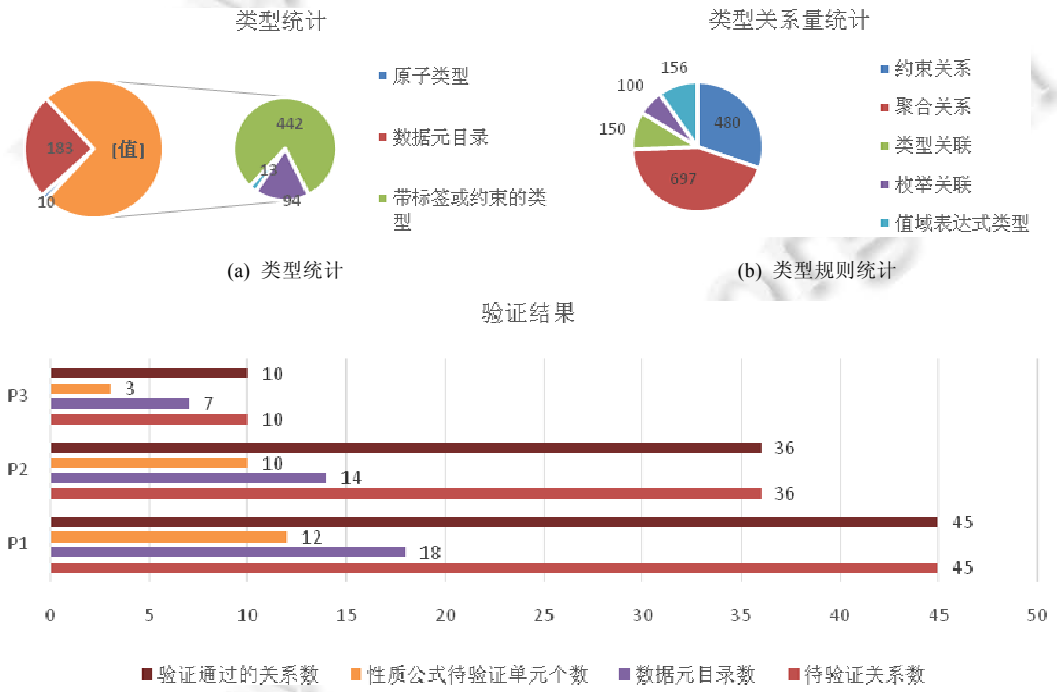
$$P_1=R_{SEQ1}\wedge(R_{SEQ21}\vee R_{SEQ22}\vee R_{SEQ23})\wedge R_{SEQ3}\wedge R_{SEQ4}\wedge R_{SEQ5}\wedge R_{SEQ6}\wedge R_{SEQ7}\wedge R_{SEQ8}\wedge R_{SEQ9}\wedge R_{SEQ10}.$$

类似地,安葬安葬业务流程和祭祀祭扫业务流程的连贯性需求 P_2 和 P_3 可以写为如下形式:

$$P_2=(R_{SEQ11}\wedge R_{SEQ12}\wedge R_{SEQ13}\wedge R_{SEQ14}\wedge R_{SEQ15}\wedge R_{SEQ16})\vee(R_{SEQ19}\wedge R_{SEQ18}\wedge R_{SEQ17})\vee R_{SEQ20};$$

$$P_3=R_{SEQ31}\vee R_{SEQ32}\vee R_{SEQ33}.$$

如图 2 所示:在验证部分,将 P_1,P_2,P_3 对应的数据元目录序列及其关系集合作为输入,采用 CheckCorrectness 算法判定其是否成立.若有一项不符合,则领域数据建模过程存在错误,即不能满足需求.经过实际验证.领域数据模型及其验证相关的结果如图 6 所示.



(a) 类型统计 (b) 类型规则统计

(c) 验证结果

Fig.6

图 6

图 6(c)列出了 P_1,P_2,P_3 等性质的验证结果.根据表 5 的定义,它们对应的性质公式待验证的单元个数分别为 12,10 和 3,待验证数据元目录关系个数分别为 45,36 和 10.容易看出,对 3 个性质验证通过的关系个数与待验证

关系个数相同.因此,本文所创建的数据模型能够满足当前业务连贯性需求.此外,根据图 6(a)和图 6(b)中的统计,殡葬领域数据模型中定义的原子类型、数据元、数据元目录以及数据元目录之间关系的个数分别为 C=10,N=549,M=184 和 L=150 个.根据表 5 和表 6,验证性质 P_1,P_2,P_3 所涉及的数据元、数据元目录和数据元目录之间关系的个数分别为 N_r=549,M_r=76 和 L_r=84.由于 $M_r<M,L_r<L$,即,本应用定义的数据元、数据元目录及其关系的个数要大于验证性质 P_1,P_2,P_3 实际所用到的相关数据元、数据元目录及其关系总数,因此,当前的领域数据模型还可以用于满足其他可能的需求.由于其需求分析方法和验证方法类似,本文不再赘述.

上述验证的基础是对任意给定的项 t,能够判定其所属的类型 T.为此,通过一个实际案例,以类型规则推导树的形式给出 DDMS 原型系统判定 $\Gamma\vdash t:T$ 的过程.对于任意输入项 t,调用 *CheckType*(t)算法,将 t 逐个分解,利用模型中已定义的类型规则及其检查算法,判定各个分项 t_i 的类型,并最终判定 t 所属的类型.本案例输入的项 t 的形式如下:

$$\begin{aligned}t=(&\{facilityId=IM\_010010\_101820091890201001\_03\_000301\},\\&\{instId=010010\_01\_000014\},\{facilitiesName=35\text{号守灵室}\},\{resourceType=06\},\\&\{buildArea=25.0\},\{capacity=2\},\{useDate=2017\text{-}04\text{-}05\},\{stopDate=2027\text{-}04\text{-}05\},\\&\{status=02\},\{facilitiesDesc=\text{守灵室用于为逝者守灵}\}).\end{aligned}$$

容易看出,t 有 10 个分项,分别记为 $t_1,\ldots,t_{10}$,则通过判定每一个 $t_i(1\leqslant i\leqslant 10)$,可得出 $t:T_{14}$ 的判定结果,如以下类型规则推导树所示:

$$\begin{aligned}&t_1:B_{272},t_2:B_1,t_3:B_{273},t_4:B_{274},t_5:B_{275},t_6:B_{276},t_7:B_{277},t_8:B_{278},t_9:B_{279},t_{10}:B_{280},\\&t=(t_1,t_2,t_3,t_4,t_5,t_6,t_7,t_8,t_9,t_{10}),\\&\frac{T_{14}=B_{272}\times B_1\times B_{273}\times B_{274}\times B_{275}\times B_{276}\times B_{277}\times B_{278}\times B_{279}\times B_{280}}{t:T_{14}}(\text{TR4}).\end{aligned}$$

限于篇幅,判定每一个 t_i 的类型规则推导树请详见附录 2 验证过程例示.

4 结 论

本文在理论和应用上的主要贡献是提出了一种基于类型理论的领域数据建模语言 DDML、领域数据建模方法 DDMM 及其验证方法和领域数据建模工具原型系统 DDMS,适用于一般领域的数据建模和验证;提出了数据元目录序列正确性验证方法,从而可以验证所创建的领域数据模型是否满足业务需求;为殡葬行业制定行业数据规范并验证了该规范的正确性,完成了大规模行业数据建模及其验证案例,说明了方法的有效性.

References:

[1] Pierce BC. Types and Programming Languages. Cambridge: MIT Press, 2002. 23−200.

[2] Constable RL. Experience using type theory as a foundation for computer science. In: Proc. of the IEEE Symp. on Logic in Computer Science. San Diego: IEEE, 1995. 266−279. [doi: 10.1109/LICS.1995.523262]

[3] Backus J. The history of Fortran I, II, and III. In: Proc. of the ACM Sigplan Notices. ACM Press, 1978. 165−180. [doi: 10.1145/800025.1198345]

[4] Martin-Lof P. Constructive mathematics and computer programming. Studies in Logic & the Foundations of Mathematics, 1982,104(1522):153−175.

[5] Liskov B, Guttag J. Program Development in Java—Abstraction, Specification, and Object-Oriented Design. Boston: Addison-Wesley Professional, 2001. 77−221.

[6] Damas L, Milner R. Principal type-schemes for functional programs. In: Proc. of the ACM Sigplan-Sigact Symp. on Principles of Programming Languages. 1982. 207−212. [doi: 10.1145/582153.582176]

[7] Hills M, Chen F, Roşu G. A rewriting logic approach to static checking of units of measurement in C. Electronic Notes in Theoretical Computer Science, 2012,290:51−67. [doi: 10.1016/j.entcs.2012.11.011]

[8] Fu C, You JY. Typing mobile resources. Ruan Jian Xue Bao/Journal of Software, 2005,16(5):979−990 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/979.htm

[9] Ma SL, Sui YF, Xu K. Well limit behaviors of term rewriting systems. Frontiers of Computer Science, 2007,1(3):283−296. [doi: 10.1007/s11704-007-0028-x]

[10] Baltazar P, Mostrous D, Vasconcelos VT. Linearly refined session types. In: Proc. of the 2nd Int'l Workshop on Linearity (EPTCS 101). Tallinn: Elsevier Science, 2012. 38−49. [doi: 10.4204/EPTCS.101.4]

[11] Caires L, Pfenning F, Toninho B. Linear logic propositions as session types. Mathematical Structures in Computer Science, 2016, 26(3):367−423. [doi: 10.1017/S0960129514000218]

[12] Budiu M, Plotkin GD. Multilinear programming with big data. Festschrift for Luca Cardelli, 2014,104(5):51−68.

[13] Chen PPS. The entity-relationship model: A basis for the enterprise view of data. In: Proc. of the National Computer Conf. Dallas: ACM Press, 1977. 77−84. [doi: 10.1109/AFIPS.1977.117]

[14] Rumbaugh J, Jacobson I, Booch G. The Unified Modeling Language Reference Manual. Reading: ADDISON-WESLEY, 2004. 3−11.

[15] Calvanese D, Lenzerini M, Nardi D. Description logics for conceptual data modeling. Fuzzy Database Modeling with XML, 1998, 10(93):3−19. [doi: 10.1007/978-1-4615-5643-5_8]

[16] Pedersen TB, Jensen CS. Multidimensional data modeling for complex data. In: Proc. of the Int'l Conf. on Data Engineering. Sydney: IEEE, 1999. 336−345. [doi: 10.1109/ICDE.1999.754949]

[17] Huang LS, Chen HP, Zheng QL, Chen GL. A new dynamic data model for object-oriented database systems. Ruan Jian Xue Bao/ Journal of Software, 2001,12(5):735−741 (in Chinese with English abstract). http://www.jos.org.cn/jos/ch/reader/view_abstract. aspx?flag=1&file_no=20010514&journal_id=jos [doi: 10.13328/j.cnki.jos.2001.05.014]

[18] Chen R, Cai XY. The meta data model based on type system. Ruan Jian Xue Bao/Journal of Software, 1995,6(5):265−275 (in Chinese with English abstract). http://www.jos.org.cn/jos/ch/reader/view_abstract. aspx?flag=1&file_no=19950502&journal_id=jos [doi: 10.13328/j.cnki.jos.1995.05.002]

[19] Wuniri QQG, Li XP, Ma SL, Lü JH. Type theory based domain data modelling and verification environment. Technical Reposrt, Vol.1, Beijing: State Key Laboratory of Software Development Environment (Beihang University), 2016. 20−35 (in Chinese).

附中文参考文献:

[8] 傅城,尤晋元.类型化移动资源.软件学报,2005,16(5):979−990. http://www.jos.org.cn/1000-9825/16/979.htm

[17] 黄刘生,陈华平,郑启龙,陈国良.一个新的面向对象数据库系统的动态数据模型.软件学报,2001,12(5):735−741. http://www.jos.org.cn/jos/ch/reader/view_abstract.aspx?flag=1&file_no=20010514&journal_id=jos [doi: 10.13328/j.cnki.jos.2001.05.014]

[18] 陈睿,蔡希尧.基于类型系统的元数据模型.软件学报,1995,6(5):265−275. http://www.jos.org.cn/jos/ch/reader/view_abstract. aspx?flag=1&file_no=19950502&journal_id=jos [doi: 10.13328/j.cnki.jos.1995.05.002]

[19] 乌尼日其其格,李小平,马世龙,吕江花.基于类型理论的领域数据建模和验证环境.技术报告,Vol.1,北京:北京航空航天大学软件开发环境国家重点实验室,2016.20−35.

附录 1

Table A1 List of data requirement

表 A1 数据需求表

需求编号	从属需求内容	需求分析
1	机构有地理信息	机构信息数据应与地理信息数据关联
2	机构有图片信息	机构信息数据应与机构图片数据关联
3	机构有法人信息	机构信息数据应与法人信息数据关联
…	…	…
27	机构有合同信息	机构信息数据应与合同信息数据关联
28	机构有服务预约信息	机构信息数据应与服务预约数据关联
29	机构有服务短信信息	机构信息数据应与服务短信数据关联
需求编号	属性需求内容	需求分析
30	每个实体数据应有唯一标识	唯一标识应符合某种表达式
31	每个实体有一般属性	例如,机构信息有名称、地点、网站等;员工有姓名、性别、出生日期等信息;逝者有死亡日期、死亡原因等信息.
35	业务数据可以跟踪	业务数据有业务标识,也应由表达式给出
36	业务数据可以有效利用实体数据	业务数据应与相应的实体数据关联

Table A1 List of data requirement (Contiuned)

表 A1 数据需求表(续)

需求编号	业务连贯性需求内容	需求分析
37	业务流程中对应一系列数据	每种业务在不同节点各有所需的数据
38	殡仪业务一次对应同一位逝者	殡仪业务流程的每条通路上的各个节点,前后应满足逝者对象关联
39	安放安葬业务一次对应同一个机构以及同一位逝者	安放安葬流程的每条通路上的各个节点,前后应满足机构以及逝者对象关联
40	祭祀祭扫业务一次对应同一个机构以及同一位逝者	殡仪业务流程的每条通路上的各个节点,前后应满足机构以及逝者对象关联

附录 2

验证过程例示:输入项 t,判断其类型 T.

假设输入项 t 为如下形式:

项 t	t=(
t_1	{*facilityId*=*IM*_010010_10182009189020100 1_03_000301},
t_2	{*instId*=010010_01_000014},
t_3	{*facilitiesName*=35 号守灵室},
t_4	{*resourceType*=06},
t_5	{*bulidArea*=25.0},
t_6	{*capacity*=2},
t_7	{*useDate*=2017-0405},
t_8	{*stopDate*=2017-0405},
t_9	{*status*=02},
t_{10}	{*facilitiesDesc*=守灵室用于为逝者守灵},
	)

验证过程:

系统通过 *CheckType*(t)进行验证.这一过程中,首先调用 *CheckComposedType*(t)方法的 *TR*4(t)分支.再根据 t 的格式对构成 t 的每个分项调用 *TR*3(t_i)(i=10).

下面给出每个 t_i 的判定过程:

(1) t_1={*facilityId*=*IM*_010010_10182009189020100 1_03_000301}的判定树如下:

$$t_{11}=IM:A_8,\ \frac{\dfrac{\dfrac{0:A_8,1:A_8}{0:A_1,1:A_1}(\mathrm{TR2})}{010010:A_1}(\mathrm{TR1}),010010\text{满足}\langle n6\rangle}{010010:A_1\text{ with }\langle n6\rangle,B_{11}=A_1\text{ with }\langle n6\rangle}(\mathrm{TR2})}{t_{12}=010010:B_{11}},$$

$$\frac{\dfrac{\dfrac{0:A_8,1:A_8,2:A_8,8:A_8,9:A_8}{0:A_1,1:A_1,2:A_1,8:A_1,9:A_1}(\mathrm{TR2})}{101008091890201001:A_1}(\mathrm{TR1}),101008091890201001\text{满足}\langle n18\rangle}{101008091890201001:A_1\text{ with }\langle n18\rangle}(\mathrm{TR2}),B_{12}=A_1\text{ with }\langle n18\rangle}{t_{13}=101008091890201001:B_{12}}(\mathrm{TR2}),$$

$$l=facilityId,\ \frac{\dfrac{\dfrac{0:A_1,6:A_1}{06:A_1}(\mathrm{TR1}),06\text{ 满足}\langle\ldots2\rangle}{06:A_1\text{ with }\langle\ldots2\rangle}(\mathrm{TR2}),B_{274}=A_1\text{ with }\langle\ldots2\rangle,\ \dfrac{\dfrac{0:A_1,1:A_1,3:A_1}{000301:A_1}(\mathrm{TR1}),000301\text{ 满足}\langle n6\rangle}{t_{15}=000301:A_1\text{ with }\langle n6\rangle}(\mathrm{TR2})}{t_{18}=06:B_{274}}(\mathrm{TR6,TR2}),B_{272}=E_7,B_{272}=\{facilityId:A_1\text{ with }\langle an38\rangle\}}{\{facilityId=IM\_010010\_10182009189020100 1\_03\_000301\}:B_{272}}(\mathrm{TR3}).$$

其中,*range*_exp(t_{11},t_{12},t_{13},t_{14},t_{15})为将 t_{11}~t_{15} 以字符“_”分隔并串连而成的字符串.

(2) t_2={*instId*=010010_01_000014}:

$$\frac{\dfrac{0:A_8,1:A_8}{0:A_1,1:A_1}(\mathrm{TR2})}{\dfrac{010010:A_1}{010010:A_1 \text{ with } \langle n6\rangle}(\mathrm{TR1}),010010\text{满足}\langle n6\rangle}(\mathrm{TR2}),B_{11}=A_1 \text{ with } \langle n6\rangle}{t_{21}=010010:B_{11}},$$

$$\frac{\dfrac{0:A_1,1:A_1}{01:A_1}(\mathrm{TR1}),01\text{ 满足}\langle ...2\rangle}{01:A_1 \text{ with } \langle ...2\rangle}(\mathrm{TR2}),B_{13}=A_1 \text{ with } \langle ...2\rangle}{t_{22}=01:B_{13}},$$

$$l=instId,\frac{\dfrac{\dfrac{0:A_1,1:A_1,4:A_1}{000014:A_1}(\mathrm{TR1}),000014\text{满足}\langle n6\rangle}{t_{15}=000014:A_1 \text{ with } \langle n6\rangle}(\mathrm{TR2})}{range\_\exp(t_{21},t_{22},t_{23}):E_3,010010\_01\_000014\text{满足}\langle an16\rangle}(\mathrm{TR6}),B_1=E_3,B_1=\{instId:A_1 \text{ with } \langle an16\rangle\}}{\{instId=010010\_01\_000014\}:B_1}(\mathrm{TR3}).$$

(3) t_1={*facilitiesName*=35 号守灵室}:

$$l=facilitiesName,\frac{\dfrac{t_{31}=35\text{号守灵室}:A_1,t_{31}\text{满足}\langle ...64\rangle}{t_{31}:A_1 \text{ with } \langle ...64\rangle}(\mathrm{TR2}),B_{273}=\{facilitiesName:A_1 \text{ with } \langle ...64\rangle\}}{\{facilitiesName=35\text{号守灵室}\}:B_{273}}(\mathrm{TR3}).$$

(4) t_4={*resourceType*=06}:

$$l=resourceType,\frac{\dfrac{t_{41}=06:A_1(\mathrm{TR1}),t_{41}\text{满足}\langle n2\rangle}{t_{91}:A_1 \text{ with } \langle n2\rangle}(\mathrm{TR2}),B_{274}=\{resourceType:A_1 \text{ with } \langle n2\rangle\}}{\{resourceType=06\}:B_{274}}(\mathrm{TR3}).$$

(5) t_5={*bulidArea*=25.0}:

$$\frac{l=buildArea,t_{51}=25.0:A_3(\mathrm{TR2}),B_{275}=\{buildArea:A_3\}}{\{buildArea=25.0\}:B_{275}}(\mathrm{TR3}).$$

(6) t_6={*capacity*=2}:

$$\frac{l=capacity,t_{61}=2:A_2(\mathrm{TR2}),B_{276}=\{capacity:A_2\}}{\{capacity=2\}:B_{276}}(\mathrm{TR3}).$$

(7) t_7={*useDate*=2017-0405}:

$$l=useDate,\frac{\dfrac{\dfrac{y_1=2,y_2=0,y_3=1,y_4=7,m_1=0,m_2=4,d_1=0,d_2=5:A_8}{t_{71}=y_1y_2y_3y_4-m_1m_2-d_1d_2=2017-04-05:A_4}(\mathrm{TR1}),t_{71}\text{满足}\langle 10\rangle}{t_{71}:A_4 \text{ with } \langle 10\rangle}(\mathrm{TR2}),B_{277}=\{useDate:A_4 \text{ with } \langle 10\rangle\}}{\{useDate=2017-04-05\}:B_{277}}(\mathrm{TR3}).$$

(8) t_8={*stopDate*=2017-0405}:

$$l=stopDate,\frac{\dfrac{\dfrac{y_1=2,y_2=0,y_3=2,y_4=7,m_1=0,m_2=4,d_1=0,d_2=5:A_8}{t_{81}=y_1y_2y_3y_4-m_1m_2-d_1d_2=2027-04-05:A_4}(\mathrm{TR1}),t_{81}\text{满足}\langle 10\rangle}{t_{81}:A_4 \text{ with } \langle 10\rangle}(\mathrm{TR2}),B_{278}=\{stopDate:A_4 \text{ with } \langle 10\rangle\}}{\{stopDate=2027-04-05\}:B_{278}}(\mathrm{TR3}).$$

(9) t_9={*status*=02}:

$$\frac{l=status,\dfrac{t_{91}=02:A_1(TR1),t_{91}\text{满足}\langle ...2\rangle}{t_{91}:A_1 \text{ with } \langle ...2\rangle}(\mathrm{TR2}),B_{279}=\{status:A_1 \text{ with } \langle ...2\rangle\}}{\{status=02\}:B_{279}}(\mathrm{TR3}).$$

(10) t_{10}={*facilitiesDesc*=守灵室用于为逝者守灵}:

$$l=facilitiesDesc,\frac{\dfrac{t_{101}=\text{守灵室用于为逝者守灵}:A_1,t_{101}\text{满足}\langle ...1024\rangle}{t_{101}:A_1 \text{ with } \langle ...1024\rangle}(\mathrm{TR2}),B_{280}=\{facilitiesDesc:A_1 \text{ with } \langle ...1024\rangle\}}{\{facilitiesDesc=\text{守灵室用于为逝者守灵}\}:B_{280}}(\mathrm{TR3}).$$

最后,根据 $t=(t_1,t_2,t_3,t_4,t_5,t_6,t_7,t_8,t_9,t_{10})$,有如下判定:

$$t_1:B_{272},t_2:B_1,t_3:B_{273},t_4:B_{274},t_5:B_{275},t_6:B_{276},t_7:B_{277},t_8:B_{278},t_9:B_{279},t_{10}:B_{280},$$
$$t=(t_1,t_2,t_3,t_4,t_5,t_6,t_7,t_8,t_9,t_{10}),$$
$$\frac{T_{14}=B_{272}\times B_1\times B_{273}\times B_{274}\times B_{275}\times B_{276}\times B_{277}\times B_{278}\times B_{279}\times B_{280}}{t:T_{14}}(\mathrm{TR4}).$$

即,输入项 t 所属的类型为 T_{14}.

乌尼日其其格(1979－),女,内蒙古赤峰人,博士生,CCF 学生会员,主要研究领域为软件形式化方法,类型系统.

李小平(1979－),男,博士,高级工程师,CCF 学生会员,主要研究领域为云计算,软件形式化方法.

马世龙(1953－),男,博士,教授,博士生导师,主要研究领域为海量信息处理的计算模型,软件工程,形式化方法.

吕江花(1975－),女,博士,副教授,CCF 专业会员,主要研究领域为软件形式化方法,软件工程,安全苛刻系统自动化测试.