计算机研究与发展
Journal of Computer Research and Development
ISSN 1000-1239/CN 11-1777/TP
42(12):2184~2191, 2005

ReDE:一个基于正则表达式的生物数据抽取方法

邓绪斌[1]　朱扬勇[2,3]

[1](浙江财经学院信息学院　杭州　310018)

[2](复旦大学计算机与信息技术系　上海　200433)

[3](上海生物信息技术研究中心　上海　201203)

(xbdeng@fudan.edu.cn)

ReDE: A Regular Expression-Based Method for Extracting Biological Data

Deng Xubin[1] and Zhu Yangyong[2,3]

[1](*School of Information*, *Zhejiang University of Finance & Economics*, *Hangzhou* 310018)

[2](*Department of Computing and Information Technology*, *Fudan University*, *Shanghai* 200433)

[3](*Shanghai Center for Bioinformation Technology*, *Shanghai* 201203)

Abstract Extracting data from heterogeneous biological data sources to build a query and analysis platform for biological scientists is currently a hot research topic. In general, data extraction process concerns many interdependent metadata. Making full use of dependencies among metadata to generate one metadata from another can reduce metadata maintenance overhead. However, many data extraction methods overlook these dependencies and require much effort to construct and maintain many metadata. In this paper, a regular expression (RE) based method named as ReDE is proposed to avoid this drawback: by building a parse tree for RE groups, an RE-based algorithm for generating relational database scheme and a general data extraction and assembling algorithm are designed. The novelty is that the RE is the only necessary metadata whose management and maintenance are relatively easy. This method can serve as the basis for building a biological database design-aiding tool and a high automatic tool for data extraction, and has been applied to extract data for the first online integrated biological data warehouse of China.

Key words biological data source; data extraction; metadata; regular expression; extraction algorithm

摘　要　从异构生物数据源抽取数据,建立查询分析平台是目前研究的热点,而抽取过程会涉及大量相互依赖的元数据,充分利用这种依赖关系可降低维护工作量. 基于正则表达式(RE)提出了 ReDE 抽取方法:通过围绕 RE 组建立分析树,设计了基于 RE 的关系数据库模式生成算法和通用抽取与组装算法,其特点是:RE 是惟一的元数据,易于管理和维护. 该方法奠定了生物数据库辅助设计工具和高自动化抽取工具的基础,已用于构建国内第 1 个整合的生物信息在线数据仓库.

关键词　生物数据源;数据抽取;元数据;正则表达式;抽取算法

中图法分类号　TP311.13

1　引　　言

整合异构生物数据源,为生物学家搭建高效的查询分析平台,已成为计算机界研究的热点[1]. 数据仓库是目前整合异构生物数据源的恰当方案之一[2]. 数据抽取是数据仓库的基础,也是本文研究的核心问题.

收稿日期:2004-06-14;修回日期:2005-08-01

基金项目:国家"八六三"高技术研究发展计划基金项目(2002AA231011);上海市重大科技基金项目(02DJ14013)

数据抽取问题可描述为给定数据源 S,确定一个 S 到数据库 R 的映射 M,该映射用数据抽取模型、抽取规则和抽取算法去抽取 S 中的数据对象,用数据库模式、映射规则和组装算法将已抽取的数据对象组装到 R 中[3,4](本文将数据抽取模型、抽取规则、数据库模式和映射规则称为元数据).

同一数据源的各种元数据间往往存在依赖关系,若能利用这种依赖关系,从一种元数据导出其他元数据,则可减少需人工维护的元数据数目.而现有抽取方法[3~6]并未有意识地利用这种依赖关系,用这些方法对具有多样性、复杂性、易变性等特征的生物数据源进行抽取时,往往需要人工构造和维护大量的元数据.

为充分利用元数据间的依赖关系,降低维护工作量,本文提出了一套称为 ReDE 的抽取方法,该方法用正则表达式(RE)作为惟一的元数据来描述生物数据源,其他元数据则通过算法导出.

ReDE 还具有进化潜力:①由于生物学家对数据库技术不熟悉,因此有必要设计一个数据库辅助设计工具.ReDE 奠定了该工具的基础.②只要能根据数据样本推断 RE,就可实现高自动的抽取工具.ReDE 也奠定了该工具的基础.

ReDE 已用于构建国内第 1 个整合的生物信息数据仓库,详见 http://www.biosino.org/biodw/.

2 相关工作

据我们所知,和 ReDE 最接近的研究工作有 DEByE[3,4]和 RoadRunner[7].DEByE[3,4]用嵌套表来描述其数据模型并将模型映射成对象关系数据库模式;ReDE 用 RE 来整合元数据,并将 RE 映射成数据模型和关系数据库模式.RoadRunner[7]的研究动机是从 HTML 样本中学习受限 RE 并映射成数据模型;ReDE 从支持复杂数据源的抽取及降低元数据维护工作量出发,考虑将非受限 RE 映射成数据模型、关系数据库模式和搜索模板.

3 正则表达式及其分析树

正则表达式(RE)是一个模式串,用于在给定主字符串中搜索与之匹配的子串.RE 中的圆括号对称为组,内层圆括号对又称子组.每个组有一个编号.为描述统一,用'('和')*'将 RE 整体括起.若')'的下一个字符为'*'、'+'或'?'则称该字符为相应组的标记,否则组的标记定义为 ε(即空串).匹配时每个组都会捕捉一个指向所匹配字符串位置的索引(记为:$[start,end]$),称为分区.所有组的分区构成一个分区数组,记为:$gp[-]$.

在设计有关 RE 的算法时通常需将 RE 映射成分析树.虽然文献[8]给出了以运算符为核心构造的分析树,但为了准确描述抽取过程,需要以组为核心构造分析树.为此先定义一些概念.

定义 1. 在 RE 中,起始于'('、')*'、')+'或')?'下一个字符,终止于'('、'|'或')'上一个字符的无圆括号非空子串称为原子.起始于'('或'|'下一个字符,终止于'|'或对应的')'上一个字符的原子和子组构成的非空串称为序列.

定义 2. 由')*'或')+'封闭的组称为多重组,反之称为单重组.

例 1. 在图 1 中,组为 $g_0,\cdots,g_6$;原子为 $a_1,\cdots,a_9$;序列为 $s_1,\cdots,s_9$;组 0~3 是多重组;组 4~6 是单重组.

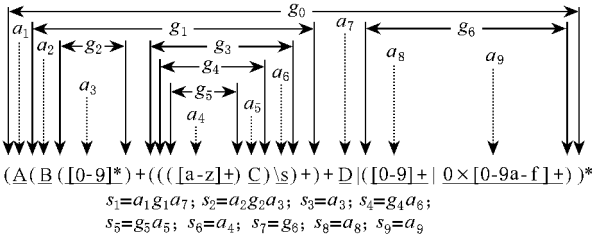


Fig. 1 Groups, atoms and sequences in RE.

图 1 RE 中的组、原子和序列

可将 RE 映射成一棵以组为核心的分析树.分析树 T 的结点集被划分为叶结点、组结点和'*'与'?'结点 3 个互不相交的结点类,且 T 的每个结点 n 都持有一个模板 $n.\Gamma$ 和一个标记 $n.\gamma\in\{\varepsilon,*,+,?\}$.这种分析树的形式定义如下:

定义 3. 由正则表达式 re 生成的分析树 $T(re)$ 是在 re 上递归地运用下列规则而获得的树:①ε 生成叶结点 $\lambda(\varepsilon)$,满足 $\lambda(\varepsilon).\gamma=\varepsilon,\lambda(\varepsilon).\Gamma=\varepsilon$;②原子 a 生成叶结点 $\lambda(a)$,满足 $\lambda(a).\gamma=\varepsilon,\lambda(a).\Gamma=a$;③序列 $s=\delta_1\cdots\delta_k(k>1)$ 生成'*'与'?'结点 $\Lambda(s)$,满足 $\Lambda(s)$的子结点 $c_1,\cdots,c_k$ 分别由 $\delta_1,\cdots,\delta_k$ 生成,$\Lambda(s).\gamma=\varepsilon,\Lambda(s).\Gamma=c_1.\Gamma\cdots c_k.\Gamma$;④组 $g=(\beta_1|\cdots|\beta_k)\tau(\tau\in\{\varepsilon,*,+,?\})$生成组结点 $\Theta(g)$,满足 $\Theta(g)$的子结点 $c_1,\cdots,c_k$ 分别由 $\beta_1,\cdots,\beta_k$ 生成,$\Theta(g).\gamma=\tau,\Theta(g).\Gamma=(c_1.\Gamma|\cdots|c_k.\Gamma)\tau$.

例 2. 图 2 给出了由图 1 中的 RE 生成的分析树.其中叶结点和原子对应,其模板标于下方;组结

点的标记标于右边(ε 省略),组编号标在其中.

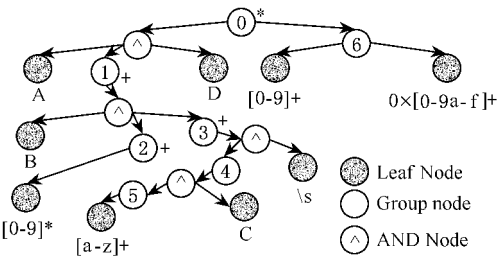


Fig. 2 Parse tree generated from the RE in Fig. 1.

图 2 由图 1 中的 RE 生成的分析树

4 从分析树到数据模型和关系数据库模式

分析树过于复杂,不宜直接用作数据模型,需要进行简化. 下面讨论具体方法.

定义 4. 标记为"+"或"*"的组结点称为表结点;所有孩子都是叶结点的组结点称为属性结点.

属性结点必被某个表结点所拥有,这个表结点就是从该属性结点出发沿分析树向上搜索所遇到的第 1 个表结点.

定义 5. 一个"与"结点是冗余的,如果它满足下列条件之一:①仅有一个子结点;②所有子结点都是"与"结点;③双亲结点是"与"结点.

算法 1. 化简分析树为数据模型.

输入:分析树 T;

输出:数据模型 T_M.

① 删除 T 中的所有叶结点,生成 T_F;

② 删除 T_F 中所有既非表又非属性的组结点,将其子结点连至其双亲,生成 T_P;

③ 除根结点外,删除 T_P 中的所有无属性表结点,将其子结点连至其双亲,生成 T_S;

④ 先序遍历 T_S,删除 T_S 中的所有冗余"与"结点,将其子结点连至其双亲,生成 T_M.

T_M 本质上是一个 XML-Schema. 虽然文献[9]给出了从 XML-Schema 到关系数据库模式的经典映射,但是这一映射不仅会自动将"*"展开而破坏组结点间逻辑关系,而且不支持"|"运算. 因此我们设计了新的映射算法 ReDE-Map.

ReDE-Map 将 T_M 中的一个表结点映射成若干个关系模式,将 T_M 中的属性结点映射成某个关系模式的属性. 为准确描述,先介绍一些记法.

表结点 $t\in T_M$ 拥有的属性结点集记为 $t.A$;t 在 T_M 中的子结点集记为 $t.C=\{c_1,c_2,\cdots,c_k\}$($t$ 无子结点时规定 $t.C=\{t\}$). 为描述方便,我们将 $t.A$ 划分成 k 个类 $t.\mathcal{D}=\{[c_1],[c_1],\cdots,[c_k]\}$($t.\mathcal{D}$ 中某些类可能为空),$[c_i]$ 类中的所有属性结点都来自以 $c_i\in t.C$ 为根的子树(称为 t 的 c_i 分枝).

ReDE-Map 算法实现下列映射规则:

(1) 表结点映射规则. 表结点 $t\in T_M$ 被映射成关系模式集 $t.\sigma=\{t.R_i\mid i=1,\cdots,k\}$($t.\sigma$ 中某些关系模式可能为空),且 $t.R_i(i=1,\cdots,k)$的属性字段都来自$[c_i]\in t.\mathcal{D}$ 类中的属性结点.

(2) 属性结点映射规则. 设属性结点 a 的拥有者为 t,且属于 $t.\mathcal{D}$ 中的$[c_i]$类,则 a 将被映射成关系模式 $t.R_i$ 的一个属性字段.

为实现键关联,规定分析树 T 的每个组结点 i 都拥有一个整型计数器 $i.ID$,计数规则为:i 的匹配被调用时归 0(根除外),i 的一个实例被发现时加 1. 因此,若设函数 $P_M(t)$返回 T_M 中离表结点 t 最近的祖先表结点,则$(t.ID,P_M(t).ID,P_M^2(t).ID,\cdots,0.ID)$可用作 t 上关系表的主键,简记为$(t.pk)$;而$(P_M(t).ID,P_M^2(t).ID,\cdots,0.ID)$则是指向父表的外键. 综合考虑上述映射规则,得下列 ReDE-Map 算法.

算法 2. ReDE-Map:将 T_M 映射成关系数据库模式.

输入:数据模型 T_M;

输出:关系数据库模式 ρ.

① 置 $\rho\leftarrow\varnothing$;
② FOR 每一个表结点 $t\in T_M$ {
③ 置 $t.\sigma\leftarrow\varnothing$;
④ FOR 每一个$[c_i]\in t.D$ {
⑤ 令 $m=|[c_i]|$;
⑥ IF $m>0$ {
⑦ $t.\sigma\leftarrow t.\sigma\cup\{t.R_i(t.pk,a_{i1},\cdots,a_{im})\mid a_{i1},\cdots,a_{im}\in[c_i]\}$;}
⑧ }
⑨ $\rho\leftarrow\rho\cup t.\sigma$;
⑩ }
⑪ RETURN (ρ).

在 $t.R_i$ 中,规定字段 $a_{i1},\cdots,a_{im}$的值取自分区的集合:$\{[start,end]\mid start\leqslant end\}$.

T_M 上临时变量的分配方法是:为属性结点 a 分配一个分区变量 $a.v$;为表结点 t 分配一个字段数等于$|t.A|$的记录变量 $t.rc$ 和$|t.\sigma|$个表变量 $t.r_i$. 抽取的数据先由 t 的属性结点送入 $t.rc$ 的相应字段中,然后,如果 t 的第 i 个子结点的匹配成功返回,t 将构造一条符合关系模式 $t.R_i$ 的记录并插入 $t.r_i$ 中.

5 ReDE 的数据抽取和组装算法

鉴于现有抽取算法[3,4,10]要么不能抽取含多重组的数据源,要么只能抽取满足苛刻条件的特殊数据源,我们设计了 ReDE-Extract 算法,描述如下:

算法 3. 数据抽取和组装算法 ReDE-Extract.

输入:组结点 N,分区 P,分析树 T,数据模型 T_M;

输出:用临时变量抽取 N 在 P 中的实例.

说明:$n.gn$ 为组 n 的编号;$Match0$ 和 $Match1$ 分别为贪心和非贪心的 RE 匹配;$strip(N.\Gamma)$ 剥去 $N.\Gamma$ 最外层括号;$Assemble$ 是实例组装函数.

① 过程 *ReDE-Extract*(group & N, partition & P) {
② IF N 不是根结点 {$N.ID \leftarrow 0$;}
③ $re \leftarrow strip(N.\Gamma)$;
④ WHILE $P.start < P.end$ AND $Match0(re, P, gp)$ {
⑤ $P.start \leftarrow gp[0].end$;/* 为抽取下一实例而修改 */
⑥ IF N 是属性结点 {$N.v \leftarrow gp[0]$;}
⑦ ELSE {/* 在分析树 T 中考察 N 的子结点 */
⑧ 设匹配来自以 N 的子结点 x 为根的分枝;
⑨ IF x 是叶结点 {转㉕;/* 无效分枝 */}
⑩ ELSE IF x 是组结点 {
⑪ $Px \leftarrow gp[0]$;
⑫ *ReDE-Extract*(x, Px);/* 递归调用 */ }
⑬ ELSE {/* x 是'与'结点 */
⑭ FOR 每一个 x 的非叶子结点 n {
⑮ *GetP* (N, n, gp);/* 计算 n 的分区 $n.P$ */ }
⑯ FOR 每一个 x 的非叶子结点 n {
⑰ *ReDE-Extract* (n, $n.P$);/* 递归调用 */}
⑱ }
⑲ }
⑳ $Assemble(N)$;
㉑ $N.ID \leftarrow N.ID + 1$;
㉒ 若 N 是根则输出 T_M 上的数据并清临时表变量;
㉓ IF N 是单重组 {转㉕;}
㉔ }
㉕ } /* *ReDE-Extract* */
㉖ 过程 *GetP*(group N, group &n, partition $gp[-]$) {
㉗ $n.P \leftarrow gp[n.gn - N.gn]$;
㉘ IF n 是多重组 {
㉙ 设 ln 为离 n 最近的非叶左兄弟结点;
㉚ IF ln 不存在 {
$n.P.start \leftarrow gp[0].start$;}
㉛ ELSE {
$n.P.start \leftarrow gp[ln.gn - N.gn].end$;}
㉜ 设 dn 为 n 的直接左兄弟结点;
㉝ IF dn 为叶结点 AND $Match1(dn.\Gamma n.\Gamma, n.P, gx)$
{$n.P.start \leftarrow gx[1].start$;}
㉟ }
㊱ } /* *GetP* */

设缓冲区的长度为 L,则算法 3 以组 0 和分区 $[0, L]$ 开始,按下列递归的方式工作:

(1) 第②~④行:初始化并计算分区数组 gp.

(2) 第⑥~⑲行:若 N 是表结点,则视匹配来源分枝的 3 种情况作处理:①匹配来自叶结点 x. 这表明遇到无效分枝,算法直接返回. ②匹配来自组结点 x. 算法递归地在 x 的分区 Px 中抽取 x 的实例. ③匹配来自'与'结点 x. 算法计算 x 的每个非叶子结点 n 的分区,然后递归地抽取 n 的实例.

(3) 第⑳行:调用 $Assemble$,将 $N.v$ 送入 T_M 上相应表结点的临时记录变量的相应字段中(若 N 是属性结点);或者构造一条符合关系模式 $N.R_x$ 的记录并插入 $N.r_x$ 中(若 N 是表结点且在 T_M 中).

6 数据抽取和组装算法分析

6.1 代价模型

给定样本 ϑ 和分析树 T,算法 3 的代价 Ω 应为 I/O 代价 C_{IO} 与搜索代价 C_S 之和,即

$$\Omega(\vartheta, T) = C_{IO}(\vartheta, T) + C_S(\vartheta, T), \quad (1)$$

C_S 就是 T 中组结点实例搜索代价之和,即

$$C_S(\vartheta, T) = \sum_{i \in G} \sum_{j \in I_i} (m_{ij} + p_{ij}), \quad (2)$$

其中,G 为 T 的组结点集;I_i 为组 i 在样本 ϑ 中的实例集;m_{ij} 和 p_{ij} 分别为第 i 组第 j 个实例的匹配代价和分区计算代价. 综合式(1)(2),代价模型为

$$\Omega(\vartheta,T)=C_{IO}(\vartheta,T)+\sum_{i\in G}\sum_{j\in I_i}(m_{ij}+p_{ij}). \quad (3)$$

6.2 算法伸缩性

宏观上看,C_{IO}远大于 C_S. 设样本大小为 S,叶结点集为 L,$l\in L$ 的匹配量为 A_l,则必有 $S=\sum_{l\in L}A_l$. 匹配时 L 中所有叶结点的匹配量都被读入内存,但并非所有叶结点的匹配量都被输出. 若用 E 表示最终输出的匹配总量(简称"抽取量")并令 $\eta=E/S$ (简称"抽取率"),则 C_{IO}可表示为

$$C_{IO}=S\times C_I+E\times C_O=(C_I+\eta\times C_O)S=(C_I/\eta+C_O)E, \quad (4)$$

其中 C_I 和 C_O 分别表示输入和输出单位数据的代价. 根据式(4)我们有下列结论.

结论 1. 当 η 不变时,算法 3 的运行代价随 S 和 E 线性增长;当 η 不太小且 C_O 远大于 C_I 时,运行代价与 E 的线性关系和 η 无关.

6.3 重叠匹配的代价

算法 3 需要在一组重叠的分区中多次调用 $Match0$ 和 $Match1$. 为讨论重叠匹配对 Ω 的影响,不失一般性,仅考虑每个组结点均只有一条分枝的情形,此时可省略所有"与"结点. 考虑下列极端情况:保持样本 ϑ 不变,在分析树 T_0 的某个以组结点 t 为根的子树上方添加 n 个串接冗余组 $z_1,\cdots,z_n$,得到分析树 T_n(如图 3 所示). 由于算法 3 用 T_0 或 T_n 去抽取 ϑ 时结果相同,所以 $C_{IO}(\vartheta,T_n)=C_{IO}(\vartheta,T_0)$. 根据式(3),$T_n$ 和 T_0 间运行时间增量为

$$\Omega_n-\Omega_0=\sum_{i=0}^{M^{(n)}}\sum_{j=1}^{N_i^{(n)}}(m_{ij}^{(n)}+p_{ij}^{(n)})-\sum_{i=0}^{M^{(0)}}\sum_{j=1}^{N_i^{(0)}}(m_{ij}^{(0)}+p_{ij}^{(0)}), \quad (5)$$

其中,Ω_n 和 Ω_0 分别表示 $\Omega(\vartheta,T_n)$ 和 $\Omega(\vartheta,T_0)$;M 表示组编号最大值;N_i 表示组 i 的实例总数.

在图 3 中,设从根到 f 路径上所有组结点构成的集合为 F,考虑到:① 冗余组 $z_1,\cdots,z_n$ 只影响 F 中组结点某些实例的搜索代价,这些实例均包含至少一个 t 实例;② T_0 和 T_n 中对应组的实例总数相同;③ $z_1,\cdots,z_n$ 的实例总数均为 N_f^t,即 f 的 N_f 个实例中包含至少一个 t 实例的实例数,故式(5)简化为

$$\Omega_n-\Omega_0=\sum_{i\in F}\sum_{j=1}^{N_i^t}(\Delta m_{ij}+\Delta p_{ij})+\sum_{i=1}^{n}\sum_{j=1}^{N_f^t}(m_{ij}+p_{ij}), \quad (6)$$

其中,Δm_{ij} 和 Δp_{ij} 分别表示 $z_1,\cdots,z_n$ 对 F 中组 i 的第 j 个实例的匹配代价和分区计算代价的影响.

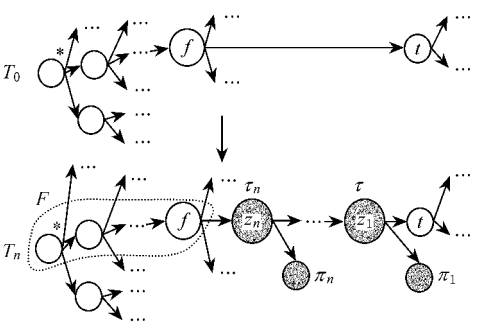


Fig. 3 Adding redundant group nodes.

图 3 添加冗余组结点

因为计算分区时仅调用非贪心的 $Match1$,故可假定 $\Delta p_{ij}=0$. 另外,由于 F 中每个结点的所有包含 t 实例的实例在匹配时都经过 $z_1,\cdots,z_n$,其匹配代价增量必和 $z_1,\cdots,z_n$ 的总匹配代价相同,即

$$\sum_{j=1}^{N_i^t}(\Delta m_{ij})=\sum_{i=1}^{n}\sum_{j=1}^{N_f^t}(m_{ij}). \quad (7)$$

因此,我们有

$$\Omega_n-\Omega_0=\sum_{i=1}^{n}\sum_{j=1}^{N_f^t}[(|F|+1)m_{ij}+p_{ij}]. \quad (8)$$

综合上述推导,Ω_n 的最终形式为

$$\Omega_n=\beta+\sum_{i=1}^{n}\lambda_i, \quad (9)$$

其中,$\beta=\Omega_0$,$\lambda_i=\sum_{j=1}^{N_f^t}[(|F|+1)m_{ij}+p_{ij}]$. λ_i 称为第 i 层重叠匹配代价增量.

当 $\pi_1=\cdots=\pi_n=\pi$ 且 $\tau_1=\cdots=\tau_n=\tau$ 时,一般有 $\lambda_i\geqslant\lambda_j$ $(i>j)$,因为冗余结点越远离 t,$Match0$ 和 $Match1$ 所需检查的组数就越多. 此时可假设 λ_i 按 $\delta>1$ 等比增长,即:$\lambda_i=\lambda_1\delta^{i-1}$,并令 $\lambda_1=\kappa$,式(9)变为

$$\Omega_n=\beta+\kappa\sum_{i=1}^{n}\delta^{i-1}=\frac{\kappa(\delta^n-1)}{\delta-1}+\beta. \quad (10)$$

为验证上述假设,可用实验数据 $\Omega_0,\Omega_1,\cdots,\Omega_n$ 来估计 κ,δ 和 β,计算公式为

$$\begin{cases}\delta=\dfrac{\sum_{i=2}^{n}(\Omega_i-\Omega_{i-1})}{\sum_{i=2}^{n}(\Omega_{i-1}-\Omega_{i-2})}=\dfrac{\Omega_n-\Omega_1}{\Omega_{n-1}-\Omega_0},\\ \kappa=\dfrac{\sum_{i=1}^{n}(\Omega_i-\Omega_{i-1})}{\sum_{i=1}^{n}\delta^{i-1}}=\dfrac{(\Omega_n-\Omega_0)(\delta-1)}{(\delta^n-1)},\\ \beta=\dfrac{1}{n+1}\sum_{i=0}^{n}\left(\Omega_i-\dfrac{\kappa(\delta^i-1)}{\delta-1}\right).\end{cases} \quad (11)$$

考虑两个极端情况:①$\pi=\varepsilon$且$\tau=\varepsilon$.此时重叠匹配代价最小;②$\pi\neq\varepsilon$且$\tau=$"+"或"*".此时重叠匹配代价最大.若分别记上述两种情况下的增长比例为$\delta_{\min}$和$\delta_{\max}$并进行推广,可得如下结论.

结论2.嵌套层数为n的路径的重叠匹配时间复杂度最坏为$O(\delta_{\max}^n)$,最好为$O(\delta_{\min}^n)$.

```
ENTRY       C00001
NAME        H2O
FORMULA     H2O
REACTION    R00001 R0002
PATHWAY     PATH:MAP00195 Ph…
ENZYME      1.1.1.22 1.1.1.23
///
ENTRY       C00002
NAME        ATP
FORMULA     C10H16N5O13P3
REACTION    R00002 R00076
PATHWAY     PATH:MAP00190 Ox…
ENZYME      1.2.1.31 1.3.99.15
///
```

(a)

```
(ENTRY\s+(.*)\n
  (NAME(\s+(.*)\n)+)?
  (FORMULA\s+(.*)\n)?
  (REACTION
    (\s+(\s?(\S+))+\n)+
  )?
  (PATHWAY
    (\s+PATH:\s+(\S{5}
      (\S{5}))\s+(.*)\n)+
  )?
  (ENZYME
    (\s+(\s?(\S+))+\n)+)?
  ///\n?
)*
```

(b)

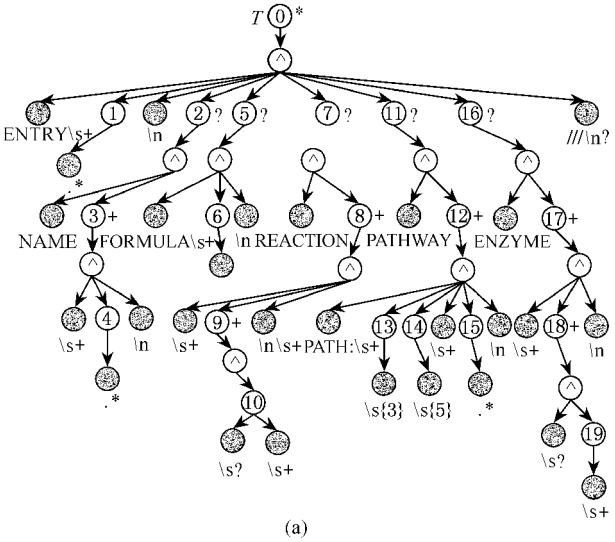
Fig. 4 Sample and corresponding RE. (a) Sample and (b) Corresponding RE

图4 数据样本和对应RE.(a)样本;(b)对应的RE

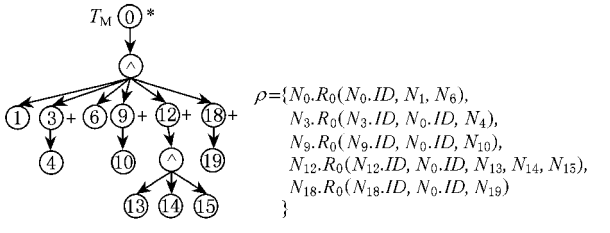
7 应用举例及实验结果

7.1 应用举例

下面举例说明ReDE的应用.图4为KEGG样本和RE;图5为T,T_M和ρ;图6为抽取结果.



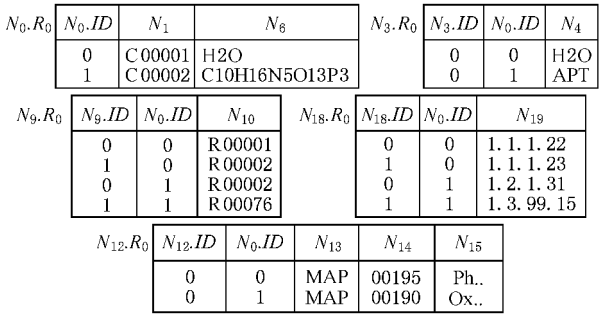
(a)



(b)

Fig. 5 Parse tree, data model and database scheme. (a) Parse tree and (b) Data model and database scheme.

图5 分析树、数据模型和数据库模式.(a)分析树;(b)数据模型和数据库模式



$N_0.R_0$

$N_0.ID$	N_1	N_6
0	C00001	H2O
1	C00002	C10H16N5O13P3

$N_3.R_0$

$N_3.ID$	$N_0.ID$	N_4
0	0	H2O
0	1	APT

$N_9.R_0$

$N_9.ID$	$N_0.ID$	N_{10}
0	0	R00001
1	0	R00002
0	1	R00002
1	1	R00076

$N_{18}.R_0$

$N_{18}.ID$	$N_0.ID$	N_{19}
0	0	1.1.1.22
1	0	1.1.1.23
0	1	1.2.1.31
1	1	1.3.99.15

$N_{12}.R_0$

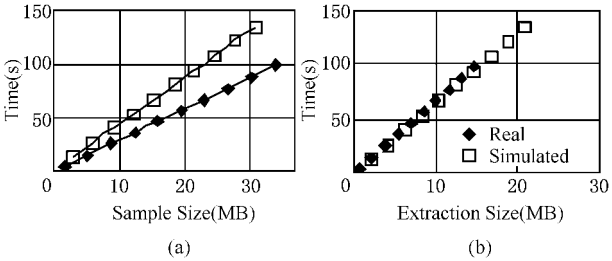
$N_{12}.ID$	$N_0.ID$	N_{13}	N_{14}	N_{15}
0	0	MAP	00195	Ph..
0	1	MAP	00190	Ox..

Fig. 6 Extraction result.

图6 抽取结果

7.2 实验结果

图7给出了在KEGG实际和模拟样本上进行的伸缩性实验结果.这一结果验证了结论1.



(a) (b)

Fig. 7 Result of scalability test. (a) Time-Sample size relationship and (b) Time-Extraction size relationship.

图7 伸缩性实验结果.(a)抽取时间—样本大小关系图;(b)抽取时间—抽取量关系图

为验证结论 2,先对图 4(b)中"REACTION"下的多重组(设为 t)添加冗余括号,方式有:①最小,即"(…(t)…)";②最大,即"(A?…(A?t)+…)+";③中等,即最小和最大方式交替进行.然后将实验点和理论曲线绘于图 8 中.图 8 的结果验证了结论 2.

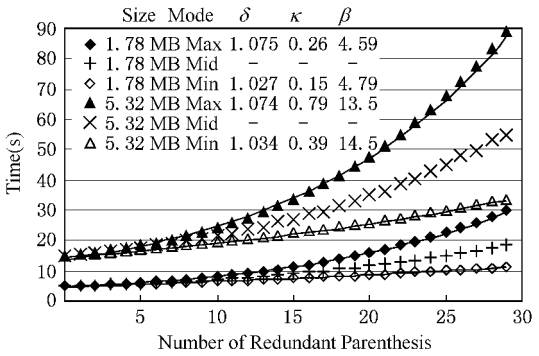


Fig. 8 Result of overlapped match test.

图 8 重叠匹配代价实验结果

8 结论和进一步的研究

本文提出了以 RE 为基础的 ReDE 抽取方法.该方法首先给出了以 RE 组为核心建立分析树的方法,然后设计了基于 RE 的关系数据库模式生成算法和通用数据抽取与组装算法,最后给出了分析抽取和组装算法伸缩性和重叠匹配时间复杂度的代价模型并验证了其基本结论. ReDE 已用于构建国内第一个整合的生物信息在线数据仓库,并为开发生物数据库辅助设计工具和高自动化抽取工具奠定了基础.进一步的工作将导入生物数据库辅助设计工具和高自动化抽取工具的研究.

参 考 文 献

1 H. Do, E. Rahm. Flexible integration of molecular-biological annotation data: The GenMapper approach. In: Proc. 9th Int'l Conf. Extending Database Technology. Berlin: Springer-Verlag, 2004. 811~822

2 S. K. Ng, L. Wong. Accomplishments and challenges in bioinformatics. IEEE IT Pro, 2004, 6(1): 12~18

3 A. H. F. Laender, A. S. da Silva, B. Ribeiro-Neto, *et al*. The Debye environment for Web data management. IEEE Internet Computing, 2002, 6(4): 60~69

4 A. H. F. Laender, B. Ribeiro-Neto, A. S. da Silva. DEByE: Data extraction by example. Data and Knowledge Engineering, 2002, 40(2): 121~154

5 B. Adelberg. NoDoSE: A tool for semi-automatically extracting structured and semistructured data from text documents. In: Proc. ACM SIGMOD Conf. Management of Data. New York: ACM Press, 1998. 283~294

6 Hu Dongdong, Meng Xiaofeng. Automatically extracting Web data using tree structure. Journal of Computer Research and Development, 2004, 41(10): 1607~1613 (in Chinese)
(胡东东,孟小峰.一种基于树结构的 Web 数据自动抽取方法.计算机研究与发展, 2004, 41(10): 1607~1613)

7 V. Crescenzi, G. Mecca, P. Merialdo. RoadRunner: Towards automatic data extraction from large Web sites. In: Proc. 27th Int'l Conf. Very Large Data Bases. San Francisco: Morgan Kaufmann, 2001. 109~118

8 C. Y. Chan, M. N. Garofalakis, R. Rastogi. RE-Tree: An efficient index structure for regular expressions. VLDB Journal, 2003, 12(2): 102~119

9 J. Shanmugasundaram, K. Tufte, C. Zhang, *et al*. Relational databases for querying XML documents: Limitations and opportunities. In: Proc. 25th Int'l Conf. Very Large Data Bases. San Francisco: Morgan Kaufmann, 1999. 302~314

10 B. Ribeiro-Neto, A. H. F. Laender, A. S. da Silva. Top-down extraction of semi-structured data. In: Proc. 6th Symposium on String Processing and Information Retrieval. Los Alamitos, CA: IEEE Computer Society Press, 1999. 176~183

Deng Xubin, born in 1964. Ph. D. and Lecturer. Received his B.S's degree from Sichuan University, in 1985, M.S's degree from Xinjiang University in 1994 and Ph. D. degree from Fudan University in 2005. His current research interests include database, data mining and bioinformatics.

邓绪斌,1964 年生,博士,讲师,主要研究方向为数据库、数据挖掘、生物信息学.

Zhu Yangyong, born in 1963. He has been professor and Ph.D. supervisor of Fudan University since 1998. His main research interests are database, knowledge base, data mining and bioinformatics.

朱扬勇,1963 年生,教授,博士生导师,主要研究方向为数据库与知识库、数据挖掘、生物信息学.

Research Background

Integrating heterogeneous biological data sources to build a convenient query and analysis platform for biological scientists has become a hot research topic. The first step to this goal is to extract data from various biological data sources. In general, the present

methods for data extraction include the following steps. First, designing proper database schemes by analyzing data sources. Next, extracting data with an algorithm driven by metadata such as data model, database schemes and mapping rules. Finally, importing extracted results into database. However, as there are evident dependencies among metadata, much effort would be demanded to maintain consistency among metadata, especially when data sources' structures change, which makes the task of building and maintaining biological database extremely hard and even unable to complete in time. In this paper, concentrating on reducing metadata maintenance overhead by making full use of dependencies among metadata, we propose a regular expression-based method named as ReDE, which specifies the structure of data source with a regular expression, and generates corresponding database scheme and mapping rules with a mapping algorithm. The novelty of ReDE is that, acting as a triple-role of the data model, the database scheme and the mapping rules, the regular expression is the only necessary metadata whose maintenance is relatively easy. Our work is supported by the National High Technology Development 863 Program of China (2002AA231011) and the Major Project of Shanghai Commission of Science & Technology (02DJ14013), and has been applied to extract data for the first online integrated biological data warehouse of China.