动态项重写计算*)

Dynamic Term Rewriting Calculus

冯 速

(北京师范大学信息科学院 北京 100875) (中国科学院计算机科学实验室 北京 100080)

Abstract Dynamic term rewriting calculus is a formal computation model for meta-computation of term rewriting systems, which has characteristic features as hierarchical declaration and dynamic rewriting, and is applied to automated formal proving for inductive theorem and termination of term rewriting systems. Here, we describe syntax, operational semantics and characteristic features of dynamic term rewriting calculus.

Keywords Dynamic term rewriting calculus, Term rewriting systems, Meta-computation, Formal computation model

1. 引言

项重写系统是一种受到广泛研究和应用的形式计算模型。一个项重写系统由一组称为重写规则的定向等式组成。它的计算基于代入、匹配和替换，除具有方向性外，与等式推导一致。虽然项重写系统形式简单、计算单纯，但它同时又具有与λ计算及图灵机相同的计算能力。正是它的简洁性及计算能力使它受到广泛的研究和应用：项重写系统为抽象数据类型提供类型、为函数型语言提供操作语义、为定理自动证明提供推理工具。对于项重写系统本身也有大量的研究：如合流性、终止性、等价性等[1~5]。

对于项重写系统的研究表明了项重写系统元计算的重要性。例如，Knuth-Bendix 的完备化算法[4]就是一种这样的元计算。为了形式地描述项重写系统的元计算，以及对于这样的形式描述进行形式验证，我们设计开发了项重写系统的元计算模型：动态项重写计算，并将其运用于归纳定理的形式自动证明、项重写系统在特定领域终止的形式自动证明等方面[2,5,6]。

动态项重写计算(简称 DTRC)的典型项是以下形式的表达式：

$[F;V:R](t)$

其中，F 和 V 分别是函数符号和变量的声明，R 是重写规则的声明，t 是被重写项。有时称这样的典型项为程序。

直观地说，子表达式$[F;V:R]$是一个声明。该声明使用 R 中的重写规则定义 F 中声明的函数。$[F;V:R](t)$是在该定义下的 t 的值。值得强调的是，不论是重写规则的两边，还是被重写项 t 均可以典型项(程序)为子项。这一特性使得 DTRC 项具有高度的层次化结构，并使我们可以进行动态重写。例如，下面是一个 DTRC 项：

```
[ base,rec,0,s;x,y;
  base(x) →x,
  rec(x,y) →s(y),
  recursive(x,y)→redundant]
([ recursive;x,y;
   recursive(0,y) →base(y),
   recursive(s(x),y)→rec(y,recursive(x,y))]
 (recursie(s(0),s(0)))
```

此项具有层次化结构，它利用一般元递归函数及其具体化操作定义由 0 和 s 构成的自然集上的加运算。

本文以上面所给的项及其变形和相应的子表达式为例概述动态项重写计算的语法和操作语义，并阐述动态项重写计算的主要特征。

2. 动态项重写计算的语法

以下是 DTRC 项及相关概念的形式定义及示例。

定义 1(DTRC 项) 设 $\mathscr{F}$ 和 $\mathscr{V}$ 分别为函数符号和变量的集合，$arity:\mathscr{F}\to\mathscr{N}$ 是 $\mathscr{F}$ 到自然数集 $\mathscr{N}$ 的映射。由 $\mathscr{F}$ 和 $\mathscr{V}$ 的元素所构成的所有项的集合$\mathbb{T}(\mathscr{F},\mathscr{V})$是满足以下条件的最小集合：

1. 若 $x\in\mathscr{V}$，则 $x\in\mathbb{T}(\mathscr{F},\mathscr{V})$；

2. 若 $t_1,\cdots,t_n\in\mathbb{T}(\mathscr{F},\mathscr{V})$，$f\in\mathscr{F}$，且 $arity(f)=n$，则 $f(t_1,\cdots,t_n)\in\mathbb{T}(\mathscr{F},\mathscr{V})$；

3. 若 $f_1,\cdots,f_m\in\mathscr{F}$，$x_1,\cdots,x_k\in\mathscr{V}$，$l_1,\cdots,l_n,r_1,\cdots,r_n,t\in\mathbb{T}(\mathscr{F},\mathscr{V})$满足：对任意的 $1\leqslant i\leqslant k$ 和 $1\leqslant j\leqslant n$，$l_j\neq x_i$ 且出现于 r_j 中的 x_i 同样出现于 l_j 中，则$[f_1,\cdots,f_l;x_1,\cdots,x_m;l_1\to r_1,\cdots,l_n\to r_n](t)\in\mathbb{T}(\mathscr{F},\mathscr{V})$。

以下假定函数符号集 $\mathscr{F}$ 和变量集 $\mathscr{V}$ 为已知，而不明确说明。

称$[F;V:R](t)$型的项为程序。其中，子表达式$[F;V:R]$称为系统，F、V 和 R 分别称为函数声明、变量声明和(重写)规则声明，t 称为被重写项。

我们可以项和系统命名为简化书写，但不容许进行递归命名。命名格式为项名≜项或系统名≜系统。

例 1 下面的 RECURSIVE 和 INSTANTIATE 是两个系统(的命名)：

```
RECURSIVE≜
    [ recursive;x,y;
      recursive(0,y) →base(y),
      recursive(s(x),y)→rec(y,recursive(x,y))]
INSTANTIATE≜
    [ base,rec,0,s;x,y;
      base(x) →x,
      rec(x,y)→s(y),
      recursive(x,y)→redundant]
```

RECURSIVE(recursive(s(0),s(0))), ADD≜INSTANTIATE(RECURSIVE(recursive(s(0),s(0))))以及 RECURSIVE(INSTANTIATE(recursive(s(0),s(0))))是 DTRC 项。

定义 2(前后文) 令□为不属于 $\mathscr{F}\cup\mathscr{V}$ 的特殊符号。前

*)国家自然科学基金(重点项目)资助项目(19931020)，教育部留学回国人员科研启动基金资助项目。冯 速 博士，副教授，主要研究方向为符号计算与自动证明。

后文$C[\![\ ]\!]$是由$\mathscr{F}$和$\mathscr{V}\cup\{\square\}$中符号构成的项。其中，$\square$仅出现于变量声明外的一个位置。使用项$t$替换前后文$C[\![\ ]\!]$中的$\square$所得的项记作$C[\![t]\!]$。

系统可以看成是特殊的前后文。

定义 3(子项) 设t和s为由$\mathscr{F}$和$\mathscr{V}$中元素组成的。若存在前后文$C[\![\ ]\!]$使得$t=C[\![s]\!]$，则称s是t的子项。

当一个子项本身是程序时，有时称其为子程序。不含子程序的项称为单纯项。由于$\mathscr{F}$和$\mathscr{V}$中元素构成的单纯项的集合记作$\mathbb{S}(\mathscr{F},\mathscr{V})$。

例 2 ADD，$R_{ECURSIVE}$(recursive(s(0)s(0)))，recursive(s(0)，s(0))，以及 redundant 等是 ADD 的子项。ADD 和$R_{ECURSIVE}$(recuresive(s(0)，s(0)))是 ADD 的子程序。recursive(s(0)，s(0))和 redundant 等是单纯项。

定义 4(有效范围) 对于项$t=[F;V:R](s)$，函数声明F的有效范围是t；变量声明V的有效范围是t的部分表达式$V:R$。

例如，对于项 ADD，外层系统$I_{NSTANTIATE}$中的函数声明的有效范围是 ADD，变量声明的有效范围是部分表达式 x，y：base(x)→x，rec(x，y)→s(y)，recursive(x，y)→redundant；内层系统$R_{ECURSIVE}$中的函数声明的有效范围是$R_{ECURSIVE}$(recursive(s(0)，s(0)))，变量声明的有效范围是部分表达式 x，y：recursive(0，y)→base(y)，recursive(s(x)，y)→rce(y，recursive(x，y))。

有效范围的概念决定下面的束缚关系。

定义 5(自由与束缚) 对于项t和函数符号f，f在t中的一个出现在t是被束缚的，当且仅当这一出现存在于t中包含f的某个函数声明的有效范围内；否则，这一出现称为t中的自由出现。同样，对于项t和变量v，v在t中的一个出现在t是被束缚的，当且仅当这一出现存在于t中包含v的某个变量声明的有效范围内；否则，这一出现称为t中的自由出现。

这里不给出“出现(occurrence)”的形式定义，而只直观地使用这一概念。形式定义见文[7]。

例 3 对于项 ADD，redundant 在 ADD 中的出现在 ADD 是自由的。base 在 ADD 中的所有出现在 ADD 中都是被束缚的。但是，虽然 base 在 ADD 的子项$R_{ECURSIVE}$(recursive(s(0)，s(0)))中出现在 ADD 中是被束缚的，它却是$R_{ECURSIVE}$(recursive(s(0)，s(0)))中的自由出现。

recursive 在$R_{ECURSIVE}$(recursive(s(0)，s(0)))中的出现在$R_{ECURSIVE}$(recursive(s(0)，s(0)))中是被束缚的，对应于这些出现在 ADD 中的出现在 ADD 中同样是被束缚的。然而，Recursive 在外层系统$I_{NSTANTIATE}$中的出现在 ADD 中则是自由的。

例 3 表明，束缚关系除与符号在项中的出现的位置有关外，与观察出现的角度也有密切的关联。束缚和自由的概念决定了 DTRC 的操作语义，它使我们既能动态地操作 DTRC 的项，又可以通过声明来控制这样的动态重写。

定义 6(框架) 设$\mathscr{F}$和$\mathscr{V}$分别为函数符号和变量的集合，$arity:\mathscr{F}\to\mathscr{N}$是$\mathscr{F}$到自然数集$\mathscr{N}$的映射。由$\mathscr{F}$和$\mathscr{F}$的元素所构成的所有框架(frames)的集合$\mathbb{F}(\mathscr{F},\mathscr{V})$是满足以下条件的最小集合：

1. 对于$\mathscr{F}$和$\mathscr{V}$中元素构成的任意的系统[F;V：R]，[F;V：R](△)$\in$F($\mathscr{F}$，$\mathscr{V}$)；

2. 对于$\mathscr{F}$和$\mathscr{V}$中元素构成的任意的系统$[F;V:R]$及任意的$fr\in\mathbb{F}(\mathscr{F},\mathscr{V})$，$[F;V:R](fr)\in\mathbb{F}(\mathscr{F},\mathscr{V})$。

其中，△是不属于$\mathscr{F}\cup\mathscr{V}$的特殊符号。

从定义可知，框架具有以下形式：

$[F_1;V_1:R_1](\cdots[F_n;V_n:R_n](\triangle)\cdots)$

上面的框架直观地表示所有型为

$[F_1;V_1:R_1](\cdots[F_n;V_n:R_n](t)\cdots)$

的项，其中，$t\in\mathbb{S}(\mathscr{F},\mathscr{V})$。

框架是项重写系统的扩展：项重写系统R对应于框架$[F;V:R](\triangle)$。其中，F是函数符号的行意有限集合；V是包含R中出现的变量的有限集合。

例 4 $R_{ECURSIVE}$(△)和$I_{NSTANTIATE}$($R_{ECURSIVE}$(△))是框架。

3. 动态项重写计算的操作语义

简略地说，在一个程序$[F;V:R](t)$中，系统$[F;V:R]$使用R中的规则定义了一个计算环境。而该程序本身代表在这一环境下的t的值。即，我们通过使用R中的规则重写t来进行计算。然而，需要注意的是，DTRC 允许层次化声明：R和t都可包含其它程序。

DTRC 的操作语义定义为$\mathbb{T}(\mathscr{F},\mathscr{V})$上的归约关系→。我们首先给出代入和匹配的定义。

定义 7(代入) 一个代入σ是一个有限序偶集$\{t_1/x_1,\cdots,t_n/x_n\}$。其中，$x_1,\cdots,x_n$为互不相同的变量，$t_1,\cdots,t_n$为项。代入$\sigma$的定义域$dom(\sigma)$定义为$\{x_1,\cdots,x_n\}$，它的值域$range(\sigma)$定义为$\{t_1,\cdots,t_n\}$。对于项$t$和代入$\sigma$，对$t$做代入$\sigma$所得的项$t\sigma$递归定义如下：

1. 若$t\in V$，则

$$t\sigma=\begin{cases}t, & \text{若 } t=x,\in dom(\sigma)\\ t & \text{否则}\end{cases};$$

2. 若$t=f(s_1,\cdots,s_m)$，则$t\sigma=f(s_1\sigma,\cdots,s_m\sigma)$；

3. 若$t=[F;V:l_1\to r_1,\cdots,l_m\to r_m](s)$，则$t\sigma=[F;V:l_1\sigma\to r_1\sigma,\cdots,l_m\sigma\to r_m\sigma](s\sigma)$

其中，$\sigma'=\{t/x\in\sigma/x\notin V\}$。

对于项t和代入$\sigma=\{t_1/x_1,\cdots,t_n/x_n\}$，$t\sigma$是通过用$t_1,\cdots,t_n$分别同时替换$t$中$x_1,\cdots,x_n$的各自由出现所得的项。

例 5 $R_{ECURSIVE}$(recursive(x，y))$\{s(0)/x,s(0)/y\}=R_{ECURSIVE}$(recursive(s(0)，s(0))))。注意，$R_{ECURSIVE}$中出现的(已声明)变量不被替换。

定义 8(匹配) 设C$[\![\ ]\!]$为前后文，s，t为项，σ为代入。这时，当且仅当$s\sigma=t$而且s中所有函数的自由出现在C$[\![s]\!]$中仍保持自由时，称s在C$[\![\ ]\!]$下通过σ与t匹配。

例 6 rec(x，y)在前后文

```
[ recursive;x,y:
  recursive(0,y) →base(y),
  recursive(s(x),y)→□]
(recursive(s(0),s(0)))
```

下，通过代入[y/x，recursive(x，y)/y]与 rec(y，recursive(x，y)))匹配(注意代入的定义域和值域中的变量的区别)。然而，recursive(x，y)在前后文$R_{ECURSIVE}$(□)下不与 recursive(s(0)，s(0))匹配；recursive 在 recursive(x，y)的出现是自由的，而在$R_{ECURSIVE}$(recursive(x，y))的出现是被束缚的。

定义 9(重写) 对于程序$[F;V:R](C[\![t]\!])$，如果存在$l\to r\in R$和代入σ使得$dom(\sigma)\subseteq V$而且l在$C[\![\ ]\!]$下通过σ与t匹配，则$[F;V:R](C[\![t]\!])$(可)重写成$[F;V:R](C[\![r\sigma]\!])$，记作$[F;V:R](C[\![t]\!])\to[F;V:R](C[\![r\sigma]\!])$。对于前后文$B[\![\ ]\!]\neq\square$和项$t,s$，若$t\to s$，则$B[\![t]\!]$(可)重写成$B$

(下转第 12 页)

在输入方式中加入个性化设计，可以从以下几条思路入手：

·输入方式的个性化选择：不同的用户可能喜欢不同的输入方式，这就要求搜索引擎能提供多种输入方式供用户选择，当然相应的检索系统应该能接受多种形式的输入。

·输入内容的个性化调整：通过用户个人描述或总结用户以往检索历史，适当调整用户的输入内容。例如：从用户以往的检索历史中得知，该用户的关键词"网络安全"实际上是指"局域网网络安全"，那么用户再次检索"网络安全"时，输入应调整为"局域网网络安全 OR 网络安全"，且第一个关键词的权重更大。

·输入界面的个性化调整：个性化调整输入界面，可以改善输入的友好程度、简化用户的输入过程、提高用户的输入速度。例如：Yahoo！分类目录可以根据用户的国籍调整输入界面的语言，还可以根据用户的地理位置、年龄段，给出不同的目录内容。

3）专用化设计也称垂直化设计　是随着用户检索领域的变化，输入方式也相应变化的一种技术，专用化输入方式可以让用户在一个专用领域内，详细、准确地表达自己的需求。例如：检索领域选择为"学术文献"，则输入方式允许用户输入题名、作者、出处等项目进行检索。又如：检索领域选择为"汽车销售"，则输入方式允许用户输入型号、厂家、功率、颜色等项目进行检索。

随着可扩展标记语言 XML 的普及，网页数据可以分门别类地表达出来，这样，专用化输入方式可以更方便地检索出准确的结果。

此外，改进输入方式还可以考虑融合人工智能、多代理等新技术，以及挖掘用户认知能力等方面。总之，提高检索质量必须建立在搜索引擎充分理解用户需求的基础上，所以在输入方式领域，我们还应进行更深入的研究。

参 考 文 献

1 Glover E J，et al. Recommending Web Documents Based on User Preferences. In：ACM SGIR 99 Workshop on Recommender Systems，Berkeley，CA，Aug. 1999

2 Cox K. Information Retrieval by Browsing. In：Proc. of The 5th Intl. Conf. on New Information Technology，1992

3 Bruza P D，Dennis，S. Query reformulation on the Internet：Empirical data and the hyperindex search engine. In：Proc. of the RIAO97 Conf. -Computer-Assisted Information Searching on the Internet，Centre de Hautes Etudes Internationales d'Informatique Documentaires，June，1997. 488～499

4 Lawrence S，Giles C L. Searching the World Wide Web. Science，1998，280(5360)：98

5 Lawrence S，Giles C L. Context and page analysis for improved web search. IEEE Internet Computing，July-August，1998. 38～46

6 Arens Y，Knoblock C A，Shen W. Query reformulation for dynamic information integration. Journal of Intelligent Information Systems，1996

7 Nastar C，Mitschke M，Meihac C. Efficient Query Refinement for Image Retrieval. IEEE Conf. on Comp. Vis. and Pattern Recognition，Santa Barbara，California，1998. 547～552

8 Jansen B J，Spink A，Bateman J，Saracevic T. Real life information retrieval：A study of user queries on the Web. ACM SIGIR Forum，1998，32(1)：5～17

（上接第 14 页）

$[\![s]\!]$），记作 $B[\![t]\!] \rightarrow B[\![s]\!]$。

定义 10（闭包与范式）　$\stackrel{\cdot}{\rightarrow}$是$\rightarrow$的自反传递闭包。对于项 s，若不存在满足 $s \rightarrow u$ 的项 u，则称 s 是一个范式。对于项 t 和 s，若 $t \stackrel{\cdot}{\rightarrow} s$ 且 s 是范式，则称 s 是 t 的一个范式。

例 7　ADD→INSTANTIATE（RECURSIVE（rec（s（0），recursive（0，s（0））））。同样，ADD→INSTANTIATE

```
([ recursive;x,y:
   recursive(0,y) →base(y),
   recursive(s(x),y)→s(recursive(x,y))]
(recursive(s(0),s(0))))
```

下面的项是 ADD 的（唯一的）范式：

```
[ base,rec,0,s;x,y:
  base(x) →x,
  rec(x,y) →s(y),
  recursive(x,y)→redundant]
([ recursive;x,y:
   recursive(0,y) →y,
   recursive(s(x),y)→s(recursive(x,y))]
 (s(s(0))))
```

4. 动态项重写计算的特征

从上述语法和操作语义的定义和示例可以看出，DTRC 具有以下主要特征：

1. 高度层次化的结构：DTRC 项中的系统嵌套呈树形结构。

2. 灵活的动态重写：可以使用上层系统中的规则重写下层系统中规则，并通过函数声明和变量声明控制这样的动态重写。

3. 项重写系统的扩展：DTRC 的框架是项重写系统的扩展。这使我们得以使用 DTRC 项形式地描述项重写系统乃至于 DTRC 框架的元计算，并对形式描述进行形式验证[2-6]；同时，由于语法和语义上的相似性，可以使用和扩展项重写系统的研究手法来研究 DTRC 的合流性、终止性等基本性质[7]。

4. 元变量的实现：可以在 DRTC 项中简单地实现在形式证明中起重要作用的元变量的概念[2-6]。这些特征及对 DTRC 的研究和应用声明，DTRC 是项重写系统的强有力的元计算模型。

参 考 文 献

1 Dershowitz N. Termination of Rewriting. J. Symb. Comput.，1987，3：69～115

2 Feng S，et al. Mechanizing Weak Termination Proving of Term Rewriting Systems by Induction. In：Proc. ICYCS'2001. 2001. 15～19

3 Huet G. Confluent Reductions：Abstract Properties and Applications to Term Rewriting Systems. J. ACM，1980，27：797～821

4 Knuth D E，Bendix P. Simple Word Problems in Universal Algebra. In：J. Leech，eds. Computational Problems in Abstract Algebra：Oxford，Pergamon Press，1970. 263～297

5 冯速．项重写系统弱基终止性的归纳证明．计算机科学，2001，28(7)：105～108

6 Feng S，et al. Mechanizing Explicit Inductive Equational Reasoning by DTRC. IEICE Trans. Inf. & System.，1995，E78-D(2)：113～121

7 Feng S，et al. Confluence Property of Simple Frames in Dynamic Term Rewriting Calculus. IEICE Trans. Inf. & Syst.，1997，E80-D(6)：625～645