通过元组解释分析最内层运行时复杂度

郭立业*
拉德堡德大学计算与信息科学研究所，荷兰
l.guo@cs.ru.nl

Deivid Vale*
拉德堡德大学计算与信息科学研究所，荷兰
deividvale@cs.ru.nl

重写中的时间复杂度通常可理解为将项归约为范式所需的步数。如何为这一度量建立复杂度界，是重写领域中一个广为人知的问题。绝大多数用于求得此类界限的技术，都是通过修改终止性证明来提取复杂度信息。例如，语义解释、递归路径序和依赖对都采用了这种做法。本文沿用这一思路，对元组解释进行调整，以用于最内复杂度分析。元组解释将项解释为元组，其中包含归约代价和范式大小的上界。与完全重写的情形不同，在最内层归约中，代价分量不必满足强单调性要求。对代价元组放宽这一要求后，我们便能证明相容性结果的最内层版本：如果项重写系统中的所有规则都能被严格定向，那么最内层重写关系就是良基的。我们还确立了元组解释能够保证相容系统具有多项式运行时复杂度界的必要条件，并描述了搜索此类解释的过程。

1 引言

在重写所导出的逐步计算模型中，时间复杂度自然地理解为到达正规形所需的重写步数。通常假定触发一个可约表达式（即执行一个计算步骤）的代价为常数。因此，低层重写实现的复杂细节（例如具体的重写引擎实现）会被忽略。只要应用规则所需的低层时间复杂度保持较低，这一假设就不会造成问题。此外，这种抽象方法的优点是，不依赖于执行相关重写系统的具体硬件平台。

在这种重写设定下，复杂度函数界定重写序列的长度，并以推导起始项的大小为参数。文献中通常考虑两种不同的复杂度概念：推导复杂度和运行时复杂度，两者的区别在于对推导初始项施加的限制不同。一方面，推导复杂度不限制初始项的集合。直观而言，它刻画了将一个项归约为范式时的最坏情况行为。另一方面，运行时复杂度要求初始项是基本初始项；从概念上说，这类项是在数据（例如整数、列表和树）上执行单次函数调用，并将数据作为参数的项。

如果程序通过重写来表示，其执行时间就与相关重写系统的运行时复杂度密切相关。同样，采用按值调用求值策略的程序也与最内层重写系统密切相关。因此，将这两个概念结合起来，我们便建立了

*作者得到 NWO VIDI 项目“CHORPE”、NWO VI.Vidi.193.075 以及 NWO TOP 项目“ICHOR”（NWO 612.001.803/7571）的资助。

P. Fontaine、D. Nantes-Sobrinho（编）：《逻辑与语义框架及其应用 2022》（LSFA’22），EPTCS 376，2023，第 34–48 页，
doi:10.4204/EPTCS.376.5

© L. Guo & D.Vale 本作品采用知识共享署名许可协议授权。

按值调用程序的代价分析与最内层项重写的运行时复杂度分析之间的联系。更重要的是，由于重写具有抽象性，我们可以不考虑任何特定的编程语言细节，仍然得出有用的项重写结果，而这些结果也可能适用于程序。关于重写在程序复杂度分析中的适用性概述，请参阅 [1, 19]。

因此，采用重写方法研究程序复杂度，使我们能够全力专注于寻找确立推导复杂度函数或运行时复杂度函数上界的技术。确定这些上界的一种自然方法，是调整用于证明终止性的证明技术，从而推导出该方法自然诱导的复杂度。许多研究都遵循这一思路。举几个例子，关于解释方法，见 [2, 4, 6, 13, 14, 20]；关于字典序和路径序，见 [5, 12, 24]；关于依赖对，见 [11, 21]。本文遵循同样的思路，着重研究应用式系统的最内层运行时复杂度是否存在上界。我们所依据的复杂度分析框架，其终止性方法是元组解释 [16]。

元组解释是解释方法的一种。因此，我们试图以某种方式解释项，使重写关系能够嵌入一个良基序中。更准确地说，我们选择一个解释域 A，它是一个集合，并配备一个定义在 A 上的良基序 $>$；然后以组合方式将项解释为 A 中的元素。对项的这种解释满足：每当发生重写，即 $s \to t$ 时，s ⟦ 和 t ⟦（分别是 s 和 t 的解释）满足 $[\![s]\!] > [\![t]\!]$。因此，项上的一个重写步骤意味着 A 上严格递减。此类解释域的良基性，加上对归约的相容性要求，保证项上不存在无限归约序列。

元组解释的定义性特征，是允许将复杂度度量拆分为代价和大小这两个抽象概念。归根结底，表达代价和大小概念所需的要素，是一个乘积 $\mathcal{C} \times \mathcal{S}$，由良基集 $\mathcal{C}$（即代价集）和拟序集 $\mathcal{S}$（即大小集）构成。直观地说，$\mathcal{C}$ 中的代价元组界定了到达范式所需的重写步数，这与前述重写代价模型一致。与此同时，$\mathcal{S}$ 中的大小元组则更为一般。我们可以用整数、实数以及项本身作为大小。按照 [16], 中的处理方式，成本–大小积的构造是依据类型的结构归纳进行的。因此，我们将每个类型 σ映射到一个成本–大小积 $\mathcal{C}_\sigma \times \mathcal{S}_\sigma$。由此，本文的一阶项形式体系遵循类型规约。

为了拓展我们这些技术的适用性，我们不仅希望给出运行时复杂度函数的上界，还希望确定其可行性的充分条件，也就是多项式上界的存在性。八十年代，Huet 和 Oppen [15] 猜想，多项式解释足以证明可行性；同一年代，Lautemann [17] 推翻了这一猜想。事实上，Hofbauer 和 Lautemann [14]的开创性工作表明，多项式解释会对推导长度导出双指数上界。对解释施加附加条件，便可恢复可行性。据我们所知，Cichon 和 Lescanne [6]最先提出了这类条件，尽管他们的研究范围仅限于数论函数。 [4], 中也证明了类似结果，作者利用对数据构造器解释的界给出了复杂性类的重写刻画。在高阶情形中也出现了相同的条件，见 [2, 16]。本文采用与 [4] 类似的方法，并表明通过对解释中的大小元组加以界定，我们可以恢复这些经典结果。

元组解释并不能提供一种完备的终止性证明方法：对于某些终止系统，无法找到相应的解释。因此，它也无法导出完备的复杂性分析框架。尽管如此，只要明智地选择代价—大小集合，它仍有望发挥强大作用。另一个局限是，一般而言，寻找解释是不可判定的；即使在多项式情形下，这一点也早已可预见 [18]。然而，不可判定性从未阻碍计算机科学家致力于将困难问题机械化。事实上，已有若干证明搜索方法

多年来，人们一直在开发自动寻找解释的方法 [3,7,8,13,25]。

贡献。 我们给出了成本–大小乘积的形式化定义（定义 1），并用它来解释定义 3 中的类型。成本–大小乘积为成本–大小元组代数提供了解释域（定义 6）。在引理 2 和 4 中，我们证明了这种方法的可靠性。在定义 5 中，我们在成本–大小乘积上引入了一个类型安全的应用算子，并证明了它具有强单调性；这是证明相容性定理1 的重要组成部分。我们在例3 中证明了 Toyama 系统的终止性，说明定理1 在我们的设定下正确刻画了最内层终止性。在引理 7 和 8 中，我们给出了实现最内层运行时复杂度可行界的充分条件。

概要。 第 2 节中，我们固定记号，并回顾重写语法的基本概念、重写复杂度的基本术语，同时介绍关于集合、序和函数的记号。第 3 节中，我们针对最内层情形调整元组解释，并证明相容性定理的最内层版本。第 4 节中，我们着手建立最内层运行时复杂度的复杂度界。第 5 节中，我们介绍寻找成本–大小元组解释的自动化技术的初步工作。最后，我们在第 6 节中总结全文。

2 预备知识

项重写系统与最内层重写。 我们考虑采用柯里化记法的简单类型一阶项重写系统。固定一个集合 $\mathcal{B}$，其元素称为基类型。集合 $\mathcal{T}_\mathcal{B}$ 类型 由文法 $\mathcal{T}_\mathcal{B} ::= \mathcal{B} \mid \mathcal{B} \Rightarrow \mathcal{T}_\mathcal{B}$ 生成。每种类型都写作 $\iota_1 \Rightarrow \cdots \Rightarrow \iota_m \Rightarrow \kappa$，其中所有 ι_i 和 κ 均为基类型。签名 是一个符号集合 $\mathcal{F}$，并配有一个元数函数 ar，该函数为每个 $\mathsf{f} \in \mathcal{F}$ 关联一种类型 $\sigma \in \mathcal{T}_\mathcal{B}$。我们称三元组 $(\mathcal{B}, \mathcal{F}, \mathrm{ar})$ 为语法签名。对于每个基类型 ι，我们假定有一个包含可数多个变量的集合 $\mathcal{X}_\iota$，并假设若 $\iota \neq \iota'$，则 $\mathcal{X}_\iota \cap \mathcal{X}_{\iota'} = \emptyset$。令 $\mathcal{X}$ 表示 $\bigcup_\iota \mathcal{X}_\iota$，并假设 $\mathcal{F} \cap \mathcal{X} = \emptyset$。

预项 $\mathbb{T}$ 的集合由文法 $\mathbb{T} ::= \mathcal{F} \mid \mathcal{X} \mid (\mathbb{T}\ \mathbb{T})$ 生成。项的集合 $\mathrm{T}(\mathcal{F}, \mathcal{X})$ 由可按如下方式定型的预项组成：(i) 若 $\mathrm{ar}(\mathsf{f}) = \sigma$，则 $\mathsf{f} : \sigma$；(ii) 若 $\mathsf{x} \in \mathcal{X}_\iota$，则 $\mathsf{x} : \iota$；(iii) 若 $\mathsf{s} : \iota \Rightarrow \tau$ 且 $\mathsf{t} : \iota$，则 $(\mathsf{s}\ \mathsf{t}) : \tau$。项的应用是左结合的，因此我们将 ((s t) u) 写作 s t u。令 vars(s) 表示 s 中出现的变量集合。基项是满足 $\mathrm{vars}(\mathsf{s}) = \emptyset$ 的项 s。若符号 $\mathsf{f} \in \mathcal{F}$ 是 s 的头符号，即 $\mathsf{s} = \mathsf{f}\,\mathsf{s}_1 \ldots \mathsf{s}_k$，则称 f 为 s 的头符号。s 的子项是项 t（记作 $\mathsf{s} \trianglerighteq \mathsf{t}$），满足：(i) $\mathsf{s} = \mathsf{t}$；或 (ii) 当 $\mathsf{s} = \mathsf{s}'\,\mathsf{s}''$ 时，t 是 s' 或 s'' 的子项。s 的真子项是一个不等于 s 的子项。替换 γ 是从变量到项的保类型映射，且集合 $\mathrm{dom}(\gamma) = \{\mathsf{x} \in \mathcal{X} \mid \gamma(\mathsf{x}) \neq \mathsf{x}\}$ 是有限的。每个替换 γ 都可扩展为从项到项的保类型映射，其在 s 上的像记作 $\mathsf{s}\gamma$，定义如下：(i) $\mathsf{f}\gamma = \mathsf{f}$；(ii) $\mathsf{x}\gamma = \gamma(\mathsf{x})$；(iii) $(\mathsf{s}\ \mathsf{t})\gamma = (\mathsf{s}\gamma)\ (\mathsf{t}\gamma)$。

项上的关系 $\to$ 若满足：对于所有类型适当的项 t 和 u，只要 $\mathsf{s} \to \mathsf{s}'$，就有 $\mathsf{t}\,\mathsf{s} \to \mathsf{t}\,\mathsf{s}'$，且 $\mathsf{s}\,\mathsf{u} \to \mathsf{s}'\,\mathsf{u}$，则称其为单调关系。重写规则 $\ell \to \mathsf{r}$ 是一对类型相同的项，满足 $\ell = \mathsf{f}\ \ell_1 \ldots \ell_k$ 且 $\mathrm{vars}(\ell) \supseteq \mathrm{vars}(\mathsf{r})$。项重写系统（TRS） $\mathcal{R}$ 是一组重写规则。由 $\mathcal{R}$ 诱导的重写关系 $\to_\mathcal{R}$ 是项上的最小单调关系，满足对于所有规则 $\ell \to \mathsf{r} \in \mathcal{R}$ 和替换 γ，$\ell\gamma \to_\mathcal{R} \mathsf{r}\gamma$。可约表达式（重写红ex）是形如 $\ell\gamma$ 的项，其中某个规则为 $\ell \to \mathsf{r}$，替换为 γ。若一个项的所有子项都不是可约表达式，则该项处于范式。若不存在无限重写序列 $\mathsf{s} \to_\mathcal{R} \mathsf{s}' \to_\mathcal{R} \mathsf{s}'' \to_\mathcal{R} \cdots$，则称 TRS $\mathcal{R}$ 是终止的。

每条重写规则 $\ell \to r$ 定义 一个符号 f，即 ℓ的头符号。对于每个 $\mathsf{f} \in \mathcal{F}$，令 $\mathcal{R}_\mathsf{f}$表示在 $\mathcal{R}$中定义 f 的重写规则集合。若符号 $\mathsf{f} \in \mathcal{F}$ 满足 $\mathcal{R}_\mathsf{f} \neq \emptyset$，则称其为定义符号；否则，

f 称为构造子。令 $\mathcal{D}$ 为定义符号集合，$\mathcal{C}$ 为构造子集合。因此 $\mathcal{F}=\mathcal{D}\cup\mathcal{C}$。数据项是形如 c $d_1\ \ldots d_k$ 的项，其中 c 是构造子，且每个d_i都是数据项。基本项是类型为 ι 且形如 f $d_1\ \ldots d_m$ 的项，其中 ι 是一个基 sort，f 是定义符号，且所有$d1,\ldots,d_m$都是数据项。我们用 $T_b(\mathcal{F})$ 表示所有基本项的集合。

例 1 我们分别将自然数和自然数列表的基类型记为自然数和列表。在下面的项重写系统中，0 : 自然数、s : 自然数 $\Rightarrow$ 自然数、nil : 列表 和 cons : 自然数 $\Rightarrow$ 列表 $\Rightarrow$ 列表 是构造子，而加法, 减法,商 : 自然数 $\Rightarrow$ 自然数 $\Rightarrow$ 自然数、append : 列表 $\Rightarrow$ 列表 $\Rightarrow$ 列表、求和 : 列表 $\Rightarrow$ 自然数 和反转 : 列表 $\Rightarrow$ 列表 是定义符号。

$$
\begin{array}{rl}
\text{加法}\ x\ 0 & \to x \\
\text{加法}\ x\ (\mathsf{s}\ y) & \to \mathsf{s}\ (\text{加法}\ x\ y) \\
\text{append nil}\ l & \to l \\
\text{append}\ (\text{cons}\ x\ q)\ l & \to \text{cons}\ x\ (\text{append}\ q\ l) \\
\text{减法}\ x\ 0 & \to x \\
\text{减法}\ 0\ y & \to 0 \\
\text{减法}\ (\mathsf{s}\ x)\ (\mathsf{s}\ y) & \to \text{减法}\ x\ y
\end{array}
\qquad
\begin{array}{rl}
\text{求和 nil} & \to 0 \\
\text{求和}\ (\text{cons}\ x\ q) & \to \text{加法}\ (\text{求和}\ q)\ x \\
\text{rev nil} & \to \text{nil} \\
\text{反转}\ (\text{cons}\ x\ q) & \to \text{append}\ (\text{反转}\ q)\ (\text{cons}\ x\ \text{nil}) \\
\text{商}\ 0\ (\mathsf{s}\ y) & \to 0 \\
\text{商}\ (\mathsf{s}\ x)\ (\mathsf{s}\ y) & \to \mathsf{s}\ (\text{商}\ (\text{减法}\ x\ y)\ (\mathsf{s}\ y))
\end{array}
$$

我们将注意力限制在最内层重写：只有不含可约真子项的可约表达式才能被规约。更准确地说，由 $\mathcal{R}$ 诱导的最内层重写关系 $\to^i_{\mathcal{R}}$ 定义如下：

(i) $\ell\gamma\to^i_{\mathcal{R}} r\gamma$ 当且仅当 $\ell\to r\in\mathcal{R}$，且 $\ell\gamma$ 的所有真子项都处于范式；(ii) $s\ t\ \to^i_{\mathcal{R}} s'\ t$ 当 $s\to^i_{\mathcal{R}} s'$ 时；以及 (iii) $s\ t\ \to^i_{\mathcal{R}} s\ t'$ 当 $t\ \to^i_{\mathcal{R}} t'$ 时。

在本文中，我们对..... 地分析最内层重写。因此，在不会产生歧义时，我们用 $\to$ 表示 $\to^i_{\mathcal{R}}$。

推导高度与复杂度。给定项上的一个良基且有限分支关系 $\to$，若存在一个长度为 n 的序列 $\mathsf{s}=\mathsf{s}_0\to\cdots\to\mathsf{s}_n=\mathsf{t}$，我们记 $\mathsf{s}\ \mathsf{n}\to\mathsf{t}$。项 s 关于 $\to$ 的推导高度 dh(s $\to$)，是以 s 开始的最长 $\to$-序列的长度，即 $\mathrm{dh}(\mathsf{s}\to)=\max\{\mathsf{n}\mid\exists\mathsf{t}\in\mathsf{T}(\mathcal{F},\mathcal{X}):\mathsf{s}\ \mathsf{n}\to\mathsf{t}\}$。项 s 的绝对大小记为 $|\mathsf{s}|$；若 s 是 $\mathcal{F}$ 中的符号或变量，则其绝对大小为 1；若 $\mathsf{s}=\mathsf{s}_1\mathsf{s}_2$，则为 $|\mathsf{s}_1|+|\mathsf{s}_2|$。为了在重写框架下表达各种复杂度概念，我们定义复杂度函数如下：$\mathrm{comp}(\mathsf{n}\to,\mathcal{T})=\max\{\mathrm{dh}(\mathsf{s}\to)\mid\mathsf{s}\in\mathcal{T}\text{且}\ |\mathsf{s}|\le\mathsf{n}\}$。直观而言，$\mathrm{comp}(\mathsf{n}\to,\mathcal{T})$ 是从 $\mathcal{T}$ 中绝对大小至多为 n 的项开始的最长 $\to$-序列的长度。我们在下表中总结四种特定情形：

	推导	运行时间
full	$\mathtt{dc}_{\mathcal{R}}(n)=\mathtt{comp}(n,\to_{\mathcal{R}},T(\mathcal{F},\mathcal{X}))$	$\mathrm{rc}_{\mathcal{R}}(n)=\mathrm{comp}(n\to_{\mathcal{R}},T_b(\mathcal{F}))$
最内层	$\mathtt{idc}_{\mathcal{R}}(n)=\mathtt{comp}(n,\to^i_{\mathcal{R}},T(\mathcal{F},\mathcal{X}))$	$\mathrm{irc}_{\mathcal{R}}(n)=\mathrm{comp}(n\to^i_{\mathcal{R}},T_b(\mathcal{F}))$

有序集与单调函数。拟序集 $(A,\sqsupseteq)$由一个非空集合 A 和定义在其上的拟序（自反且传递）$\sqsupseteq$ 组成。扩展良基集 $(A,>,\ge)$是一个非空集合A，并配备一个良基序 $>$和定义在A 上的一个拟序 $\ge$，使得 $\ge$ 与 $>$相容，即$x>y$ 蕴含$x\ge y$，且$x>y\ge z$ 蕴含$x>z$。下文将扩展良基集简称为良基集。

给定拟序集 (A $\sqsupseteq$) 和 (B $\sqsupseteq$)，若 x $\sqsupseteq$ y 蕴含 f (x) $\sqsupseteq$ f (y)，则称函数 f : A $\longrightarrow$ B 为弱单调函数。用 A $\Longrightarrow$ B 表示从 A 到 B 的弱单调函数集合。B 上的比较算子 $\sqsupseteq$ 在 A $\Longrightarrow$ B 上诱导出如下逐点比较：若对所有 x $\in$ A，都有 f (x) $\sqsupseteq$ g(x)，则 f $\sqsupseteq$ g。因此，(A $\Longrightarrow$ B $\sqsupseteq$) 也是一个拟序集。给定良基集 (A $>, \geq$) 和 (B $>, \geq$)，若 x $>$ y 蕴含 f (x) $>$ f (y)，且 x $\geq$ y 蕴含 f (x) $\geq$ f (y)，则称函数 f : A $\longrightarrow$ B 为强单调函数。

3 元组解释

本节将在最内层重写的语境下介绍元组代数的概念。我们首先将类型解释为成本–大小乘积，再将项解释为成本–大小元组，最后证明相容性定理的最内层版本。

3.1 类型作为成本–规模积

我们首先为 $\mathcal{T}_{\mathcal{B}}$ 中的类型构造成本–大小指称语义。我们的目标是定义一个函数$(\!|\cdot|\!)$，它将每种类型 $\sigma \in \mathcal{T}_{\mathcal{B}}$映射到一个良基集 $(\!|\sigma|\!)$，即 σ的成本–大小解释。

定义 1（成本–大小乘积）给定一个良基集 $(\mathcal{C}, >, \geq)$，称为代价集，以及一个拟序集 $(\mathcal{S}, \sqsupseteq)$，称为大小集，我们将 $\mathcal{C} \times \mathcal{S}$称为 $(\mathcal{C}, >, \geq)$ 与 $(\mathcal{S}, \sqsupseteq)$ 的成本–大小乘积，其元素称为成本–大小元组。

给定一个成本–规模积 $\mathcal{C} \times \mathcal{S}$，$\mathcal{C}$ 的良基性和 $\mathcal{S}$ 上的拟序自然地在笛卡尔积 $\mathcal{C} \times \mathcal{S}$ 上诱导出如下的序结构。

定义 2（乘积序）设 $(\mathcal{C}, >, \geq) \times (\mathcal{S}, \sqsupseteq)$ 是一个成本–规模积。则我们定义$\succ, \succcurlyeq$上的关系如下：对所有 $\langle x, y\rangle$ 以及 $\langle x', y'\rangle$，若它们属于 $\mathcal{C} \times \mathcal{S}$，

(i) $\langle x, y\rangle \succ \langle x', y'\rangle$，当 $x > x'$ 且 $y \sqsupseteq y'$ 时；

以及 (ii) $\langle x, y\rangle \succcurlyeq \langle x', y'\rangle$，当 $x \geq x'$ 且 $y \sqsupseteq y'$ 时。

接下来，我们证明按上述方式排序的成本–规模积构成一个良基集。

引理 1 三元组 $(\mathcal{C} \times \mathcal{S}, \succ, \succcurlyeq)$ **是一个良基集。**

证明 由定义 1 可立即得出，$\succ, \succcurlyeq$ 是传递的，且 $\succcurlyeq$ 是自反的。为了证明$\succ$ 是良基的，注意到若存在 $\langle x_1, y_1\rangle \succ \langle x_2, y_2\rangle \succ \cdots$，则会推出 $x_1 > x_2 > \cdots$，这不可能成立，因为 $>$是良基的。

我们仍需检查 $\succcurlyeq$ 是否与 $\succ$相容。

- 假设 $\langle x, y\rangle \succ \langle x', y'\rangle$。由于 $x > x'$蕴含 $x \geq x'$，因此有 $\langle x, y\rangle \succcurlyeq \langle x', y'\rangle$。
- 假设$\langle x,\ y\rangle \succ \langle x',\ y'\rangle \succcurlyeq \langle x'',\ y''\rangle$。由于 $x > x' \geq x''$ 蕴含 $x > x''$，且 $\sqsupseteq$ 具有传递性，因此有$\langle x,\ y\rangle \succ \langle x'',\ y''\rangle$。 ■

现在，我们将类型解释为一种特殊的成本–大小乘积。

定义 3（类型的解释）令 $\mathcal{B}$ 表示基 sort 集。一个用于 $\mathcal{B}$的解释键 $\mathcal{J}_{\mathcal{B}}$ 将每个基 sort ι 映射到一个带有最小元的拟序集（$\mathcal{J}_{\mathcal{B}}(\iota)$ $\sqsupseteq$）。对于每个类型 $\sigma \in \mathcal{T}_{\mathcal{B}}$，我们将 σ的成本–大小解释定义为乘积 $(\!|\sigma|\!) = \mathcal{C}_\sigma \times \mathcal{S}_\sigma$，其中

$$\mathcal{C}_\sigma = \mathbb{N} \times \mathcal{F}^{\mathsf{c}}_\sigma$$

$$\mathcal{F}^{\mathsf{c}}_\iota = \mathtt{unit} \qquad\qquad \mathcal{S}_\iota = \mathcal{J}_{\mathcal{B}}(\iota)$$

$$\mathcal{F}^{\mathsf{c}}_{\iota \Rightarrow \tau} = \mathcal{S}_\iota \Longrightarrow \mathcal{C}_\tau \qquad\qquad \mathcal{S}_{\iota \Rightarrow \tau} = \mathcal{S}_\iota \Longrightarrow \mathcal{S}_\tau$$

其中，单位元 $=\{\mathfrak{U}\}$ 在 $\geq$ 下构成拟序，且 $\mathfrak{U}\geq\mathfrak{U}$。所有 $\mathcal{F}^{\mathsf{c}}_{\iota\Rightarrow\tau}$ 和 $\mathcal{S}_{\iota\Rightarrow\tau}$ 均按逐点比较排序。集合 $\mathcal{C}_\sigma$ 按如下方式排序：$(n,\ f)>(m,\ g)$，当且仅当 $n>m$ 且 $f\geq g$；$(n,\ f)\geq(m,\ g)$，当 $n\geq m$ 且 $f\geq g$ 时成立。该定义要求所有（$\mathcal{C}_\sigma$，$\geq$）和（$\mathcal{S}_\sigma$，$\sqsupseteq$）都是拟序集，这由以下引理保证。

引理 2 对于任意类型 σ，$(\mathcal{C}_\sigma,>,\geq)$ 都是良基集，而 $(\mathcal{S}_\sigma,\sqsupseteq)$ 是带有最小元的拟序集。因此，$(\!|\sigma|\!)$ 是一个成本–规模积。

证明 当 σ 是一个基 sort 时，$\mathcal{C}_\sigma=\mathbb{N}\times$ 单位 $\cong\mathbb{N}$ 且 $\mathcal{S}_\sigma=\mathcal{J}_{\mathcal{B}}(\sigma)$，因此该命题显然成立。当 $\sigma=\iota\Rightarrow\tau$ 时，我们有 $\mathcal{C}_\sigma=\mathbb{N}\times\mathcal{F}^{\mathsf{c}}_{\iota\Rightarrow\tau}$，$\mathcal{F}\mathsf{c}_{\iota\Rightarrow\tau}=\mathcal{J}_{\mathcal{B}}(\iota)\Longrightarrow\mathcal{C}_\tau$ 且 $\mathcal{S}_\sigma=\mathcal{J}_{\mathcal{B}}(\iota)\Longrightarrow\mathcal{S}_\tau$。根据归纳法，$(\mathcal{C}_\tau,\geq)$ 和 $(\mathcal{S}_\tau,\sqsupseteq)$ 是拟序集。$(\mathcal{F}^{\mathsf{c}}_{\iota\Rightarrow\tau},\geq)$ 和 $(\mathcal{S}_\sigma,\sqsupseteq)$ 也是拟序集，它们按照逐点比较排序。根据引理 1，$(\mathcal{C}_\sigma,>,\geq)$ 是一个良基集。$(\mathcal{S}_\sigma,\sqsupseteq)$ 的一个最小元是常值函数 $\boldsymbol{\lambda}\mathrm{x}.\bot$，其中 $\bot$ 是 $(\mathcal{S}_\tau,\sqsupseteq)$ 的最小元。■

代价分量 $\mathcal{C}_\sigma$ of $(\!|\sigma|\!)$ 包含了将类型为 σ 的项约化到其范式所需代价的信息。它由两部分组成：一部分是数值，另一部分是函数。函数部分 $\mathcal{F}^{\mathsf{c}}_\sigma$ 在 σ 只是一个基 sort 时退化为单位；而当 $\sigma=\iota\Rightarrow\tau$ 是函数类型时，它确实是一个函数空间。后一种情况下，$\mathcal{F}^{\mathsf{c}}_\sigma=\mathcal{S}_\iota\Longrightarrow\mathcal{C}_\tau$ 由定义域为 $\mathcal{S}_\iota$ 的弱单调函数构成，其中 $\mathcal{S}_\iota$ 是 $(\!|\iota|\!)$ 的大小分量。这与基于图灵机的标准复杂度概念非常一致：在该概念中，时间复杂度以输入大小为参数。

为了使用定义 3 解释类型，我们需要一个具体的解释键。在我们的例子中，一类特定的解释键会将每个基 sort ι 映射到形如 $(\mathbb{N}^{K[\iota]},\sqsupseteq)$ 的大小集，其中 $K[\iota]\geq 1$，并按如下方式排序：$\langle x1,\ldots,x_{K[\iota]}\rangle\sqsupseteq\langle y1,\ldots,y_{K[\iota]}\rangle$，若对所有 i，均有 $x_i\geq y_i$。除非另有说明，否则我们都使用这一类解释键。我们采用语义方法（参见 [16]）来确定每个基 sort ι 对应的数值 $K[\iota]$。例如，nat 是一元表示的自然数基 sort，因此数字 $n\in\mathbb{N}$ 由数据项 s (...(s 0)) 表示，也就是将 s 连续应用于 0，共 n 次。由此可见，此类项中 s 的出现次数是衡量其大小的合理指标，因此我们令 $K[\mathrm{nat}]=1$。另一个例子是 list。要刻画列表的大小，除了列表长度外，我们可能还需要了解各个元素的信息。因此，我们同时记录列表长度和其中元素的最大大小。这样一来 $K[\mathrm{list}]=2$。在例 2 中，我们依照这一思路解释 nat,list 构造子。

定义 4 $(\!|\sigma|\!)$ 中的成本–大小元组写作 $\langle(n,\ f_{\mathsf{c}}),\ f_{\mathsf{s}}\ \rangle$，其中 $n\in\mathbb{N}$、$f_{\mathsf{c}}\ \in\mathcal{F}^{\mathsf{c}}_\sigma$，且 $f_{\mathsf{s}}\ \in\mathcal{S}_\sigma$。当 σ 是函数类型时，我们将 f_{c} 称为成本函数，并将 f^{s} 称为大小函数。

为了定义项的解释（定义 7），我们需要为成本–大小元组定义一种应用。更准确地说，给定 $fff\ \in(\!|\iota\Rightarrow\tau|\!)$ 和 $xxx\in(\!|\iota|\!)$，我们的目标是定义 $fff\ \cdot xxx\in(\!|\tau|\!)$，即将 fff 应用于 xxx。下面用一个具体例子说明这种应用应如何进行。考虑例 1 中的函数 append : 列表 $\Rightarrow$ 列表 $\Rightarrow$ 列表。它以两个列表 q 和 l 为输入。append 的预期成本–大小指称语义是一个元组 $fff\ =\langle(n,f^{\mathsf{c}}),f^{\mathsf{s}}\rangle\in(\!|$列表 $\Rightarrow$ 列表 $\Rightarrow$ 列表$|\!)$，其中

$$
\begin{aligned}
&n\in\mathbb{N},\\
&f^{\mathsf{c}}\in\overbrace{\mathcal{S}_{\mathsf{list}}}^{\text{size of } q}\Longrightarrow(\mathbb{N}\times(\overbrace{\mathcal{S}_{\mathsf{list}}}^{\text{size of } l}\Longrightarrow(\mathbb{N}\times\mathtt{unit}))),\text{ and}\\
&f^{\mathsf{s}}\in\overbrace{\mathcal{S}_{\mathsf{list}}}^{\text{size of } q}\Longrightarrow(\overbrace{\mathcal{S}_{\mathsf{list}}}^{\text{size of } l}\Longrightarrow\mathcal{S}_{\mathsf{list}}).
\end{aligned}
$$

对于第一个列表 q，取一个来自 ⟦列表⟧的成本–大小元组 $\mathrm{xxx} = \langle(\mathrm{m}\ \mathfrak{U}), \mathrm{x}^{\mathsf{s}}\rangle$。我们将 f c 和 f s 分别应用于 x^{s}，得到 $\mathrm{f\ c}(\mathrm{x}^{\mathsf{s}}) = (\mathrm{k}, \mathrm{h}) \in \mathbb{N} \times (\mathcal{S}_{\mathsf{list}} \Longrightarrow (\mathbb{N} \times \mathrm{unit}))$ 和 $\mathrm{f\ s}(\mathrm{x}^{\mathsf{s}}) \in \mathcal{S}_{\mathsf{list}} \Longrightarrow \mathcal{S}_{\mathsf{list}}$。然后将数值部分相加，并将所有数据汇集到新的成本–大小元组 $\langle(\mathrm{n} + \mathrm{m} + \mathrm{k}, \mathrm{h}), \mathrm{f\ s}(\mathrm{x}^{\mathsf{s}})\rangle$ 中。以下定义总结了这一过程。

定义 5（语义应用） 给定$\mathit{fff} = \langle(n, f_{\mathsf{c}}), f_{\mathsf{s}}\rangle \in \llparenthesis \iota \Rightarrow \tau \rrparenthesis$和$\mathit{xxx} = \langle(m\ \mathfrak{U}), x^{\mathsf{s}}\rangle \in \llparenthesis \iota \rrparenthesis$，则$\mathit{fff}$对$\mathit{xxx}$的语义应用，记作$\mathit{fff} \cdot \mathit{xxx}$，为 $\langle(n + m + k, h), f_{\mathsf{s}}(x^{\mathsf{s}})\rangle$ 其中$f_{\mathsf{c}}(x^{\mathsf{s}}) = (k, h)$。

语义应用满足左结合律，因此 $\mathit{fff} \cdot \mathit{ggg} \cdot \mathit{hhh}$ 表示 $(\mathit{fff} \cdot \mathit{ggg}) \cdot \mathit{hhh}$。此定义符合类型，具体由下面的引理陈述。

引理 3 若 $\mathit{fff} \in \llparenthesis \iota \Rightarrow \tau \rrparenthesis$ 且 $\mathit{xxx} \in \llparenthesis \iota \rrparenthesis$，则 $\mathit{fff} \cdot \mathit{xxx} \in \llparenthesis \tau \rrparenthesis$。

注 1 由于 $\mathbb{N} \times \mathrm{unit}$ 与 $\mathbb{N}$序同构，因此除非另有说明，我们将 $\mathbb{N} \times \mathrm{unit}$ 与 $\mathbb{N}$等同，并将 $(m\ \mathfrak{U})$ 与 m等同。因此，我们将 $\langle m, x^{\mathsf{s}}\rangle$ 写作 $\llparenthesis \iota \rrparenthesis$ 中的成本–大小元组，其中 ι 是一个基 sort。

3.2 成本–大小元组代数

语法签名 $(\mathcal{B}, \mathcal{F}, \mathrm{ar})$ 的一个解释，将 $\mathcal{T}_{\mathcal{B}}$中的类型以及每个 $\mathrm{f} : \sigma \in \mathcal{F}$解释为 $\llparenthesis \sigma \rrparenthesis$中的一个元素。下方的定义对此作了形式化说明。

定义 6 语法签名 $(\mathcal{B}, \mathcal{F}, \mathrm{ar})$ 上的一个代价–大小元组代数 $(\llparenthesis \cdot \rrparenthesis, \mathcal{J})$由以下部分组成：

(i) 一个成本–大小积族 $\{\llparenthesis \sigma \rrparenthesis\}_{\sigma \in \mathcal{T}_{\mathcal{B}}}$，以及

(ii) 一个解释 解释函数 $\mathcal{J} \colon \mathcal{F} \longrightarrow \biguplus_{\sigma} \llparenthesis \sigma \rrparenthesis$，它为每个 $\mathrm{f} : \sigma$关联一个元素 $\mathcal{J}_{\mathrm{f}} \in \llparenthesis \sigma \rrparenthesis$.

我们将解释的概念扩展到项，并使用赋值将类型为 ι 的变量映射到 $\llparenthesis \iota \rrparenthesis$ 中的元素。在最内层重写中，我们假设变量没有代价。

定义 7 固定一个代价–大小元组代数 $(\llparenthesis \cdot \rrparenthesis, \mathcal{J})$。赋值 $\alpha \colon \mathcal{X} \longrightarrow \biguplus_{\sigma} \llparenthesis \sigma \rrparenthesis$ 是一个函数，它将每个变量 $x \colon \iota$ 映射为一个零代价元组 $\langle 0, x^{\mathsf{s}}\rangle \in \llparenthesis \iota \rrparenthesis$。项 s 在赋值 α 下的解释，记作 $[\![s]\!]^{\mathcal{J}}_{\alpha}$，定义如下：

$$[\![\mathrm{f}]\!]^{\mathcal{J}}_{\alpha} = \mathcal{J}_{\mathrm{f}} \qquad [\![x]\!]^{\mathcal{J}}_{\alpha} = \alpha(x) \qquad [\![s\ t]\!]^{\mathcal{J}}_{\alpha} = [\![s]\!]^{\mathcal{J}}_{\alpha} \cdot [\![t]\!]^{\mathcal{J}}_{\alpha}$$

我们记作 $[\![s]\!]$，而不记作 $[\![s]\!]^{\mathcal{J}}_{\alpha}$，只要 α 和 $\mathcal{J}$ 是全称量化的，或可由上下文清楚得知。在这两种情况下，我们都可以记作 $[\![x]\!] = x$，而不记作 $[\![x]\!]^{\mathcal{J}}_{\alpha} = \alpha(x)$。作为引理 3的推论，项的解释符合类型。

引理 4 若 $s : \sigma$，则 $[\![s]\!] \in \llparenthesis \sigma \rrparenthesis$。

Let σ设 $\iota_1 \Rightarrow \ldots \Rightarrow \iota_m \Rightarrow \kappa$，其中所有 ι_i和 κ都是基 sort。$\mathcal{C}_{\sigma}$的元素可写为

$$\begin{aligned}(e_0, &\boldsymbol{\lambda} x_1.\\ &(e_1, \boldsymbol{\lambda} x_2.\\ &\quad \ldots\\ &\quad (e_{m-1}, \boldsymbol{\lambda} x_m.\\ &\qquad (e_m, \mathfrak{U}))\ldots)).\end{aligned} \tag{1}$$

当 $e_0 = e_1 = \cdots = e_{m-1} = 0$ 时，我们将 $(\boldsymbol{\lambda} x_1 \ldots x_m . e_m)$ 作为简写。

例 2 设 $\mathcal{S}_{\mathsf{nat}}$ 和 $\mathcal{S}_{\mathsf{list}}$ 分别是 $\mathbb{N}$ 和 $\mathbb{N}\times\mathbb{N}$。回顾一下，自然数的大小是 s 出现的次数，而列表的大小是一对 $q=(q_{\mathsf{l}},\ q_{\mathsf{m}})$，其中 q_{l} 是长度，q_{m} 是元素的最大大小。我们将构造子解释如下：

$$\mathcal{J}_0=\langle 0,0\rangle \qquad \mathcal{J}_{\mathsf{s}}=\langle(\boldsymbol{\lambda}x.0),\boldsymbol{\lambda}x.x+1\rangle$$
$$\mathcal{J}_{\mathsf{nil}}=\langle 0,(0,0)\rangle \qquad \mathcal{J}_{\mathsf{cons}}=\langle(\boldsymbol{\lambda}xq.0),\boldsymbol{\lambda}xq.(q_{\mathsf{l}}+1,\max(x,q_{\mathsf{m}}))\rangle$$

0 和 nil 都没有代价，因为它们是没有函数类型的构造子。采用最内层重写时，具有函数类型的构造子（例如 s 和 cons）具有$e_0=\cdots=e_m=0$形式 (1) 的代价。

注2 在定义 7 中，我们要求赋值将变量解释为零代价元组。这是一项重要但微妙的要求，只有在采用最内层归约时才成立。确实，如果归约不受限制，我们就可以将规则左侧的变量替换为包含可约表达式的项，而这些可约表达式的代价理应计入。因此，在完全重写中不计算变量的代价会导致分析不可靠。此外，零代价元组使我们能够证明项重写系统 $\mathcal{R}$ 在例3中的最内层终止性，而该系统在完全重写下并不终止。

3.3 相容性定理

粗略地说，相容性定理（定理1）指出，如果 $\mathcal{R}$ 与元组代数 $\mathcal{A}$相容，那么最内层重写关系 $\to^{i}_{\mathcal{R}}$ 就嵌入到成本–大小乘积上的良基序中。接下来的两个引理是证明该定理所需的技术性结果。引理5指出解释在替换下封闭，而引理6则为语义应用提供强单调性。

定义8 固定一个代价–大小元组代数 $(\!|\cdot|\!),\mathcal{J})$。替换 γ在赋值 α 下是零代价的，当且仅当对于每个变量x，$[\![\gamma(x)]\!]^{\mathcal{J}}_{\alpha}$ 都是零代价元组。因此，给定赋值 α 和零代价替换 γ，函数 $\alpha^{\gamma}=[\![\cdot]\!]^{\mathcal{J}}_{\alpha}\circ\gamma=[\![\gamma(\cdot)]\!]^{\mathcal{J}}_{\alpha}$也是一个赋值。

引理5（替换）如果 γ是在赋值 α下的零代价替换，那么对于任意项 s，$[\![s\gamma]\!]^{\mathcal{J}}_{\alpha}=[\![s]\!]^{\mathcal{J}}_{\alpha^{\gamma}}$。

引理6应用泛函$\mathtt{App}(\mathit{fff},\mathit{xxx})=\mathit{fff}\ \cdot\mathit{xxx}$ 对两个参数都强单调。

证明 我们需要证明：(i) 若 fff $\succ$ ggg 且 xxx $\succcurlyeq$ yyy，则 App(fff , xxx) $\succ$ App(ggg, yyy)；(ii) 若 fff $\succcurlyeq$ ggg 且 xxx $\succ$ yyy，则 App(fff ,xxx) $\succ$ App(ggg, yyy)；(iii) 若 fff $\succcurlyeq$ ggg 且 xxx $\succcurlyeq$ yyy，则 App(fff ,xxx) $\succcurlyeq$ App(ggg, yyy)。考虑成本–大小元组 fff , ggg $\in(\!|\iota\Rightarrow\tau|\!)$ 和 xxx,yyy $\in(\!|\iota|\!)$。令 fff $=\langle$(n, f c), f s $\rangle$, ggg $=\langle$(m, $\mathsf{g^c}$), $\mathsf{g^s}\rangle$, xxx $=\langle\mathsf{x^c},\mathsf{x^s}\rangle$, 且 yyy $=\langle\mathsf{y^c},\mathsf{y^s}\rangle$。我们先证明 (i)，并注意到 (ii) 和 (iii) 可用类似的推理得出。事实上，若 fff $\succ$ggg 且 xxx $\succcurlyeq$yyy，则有 n $>$m、f c $\geq\mathsf{g^c}$、f s $\sqsupseteq\mathsf{g^s}$、$\mathsf{x^c}\geq\mathsf{y^c}$以及 $\mathsf{x^s}\sqsupseteq\mathsf{y^s}$。因此，令 f c($\mathsf{x^s}$) =(k, h) 且 $\mathsf{g^c}(\mathsf{y^s})$ =(k′, h′)，可得：

$$\text{应用}(\mathit{fff},\ \mathit{xxx})=\langle\ (n,f^{\mathsf{c}}),f^{\mathsf{s}}\rangle\cdot\langle x^{\mathsf{c}},x^{\mathsf{s}}\rangle=\langle(n+x^{\mathsf{c}}+k,h),f^{\mathsf{s}}(x^{\mathsf{s}})\rangle>\left\langle(m+y^{\mathsf{c}}+k',h'),g^{\mathsf{s}}(y^{\mathsf{s}})\right\rangle=\mathtt{App}(\boldsymbol{g},\boldsymbol{y})$$

■

定义9 若项重写系统 $\mathcal{R}$ 与代价–大小元组代数 $(\!|\cdot|\!),\mathcal{J})$ 相容，则 $[\![\ell]\!]^{\mathcal{J}}_{\alpha}\succ[\![r]\!]^{\mathcal{J}}_{\alpha}$对所有规则 $\ell\to r\in\mathcal{R}$ 及赋值 α均成立。

定理1（相容性）设 $\mathcal{R}$ 是一个与成本–大小元组代数$(\!|\cdot|\!),\mathcal{J})$相容的项重写系统。于是，对于任意一对项 s 和 t，只要 $s\to^{i}_{\mathcal{R}}t$，就有 $[\![s]\!]^{\mathcal{J}}_{\alpha}\succ[\![t]\!]^{\mathcal{J}}_{\alpha}$。

证明 我们对 $\to^i_{\mathcal{R}}$使用归纳法。对于基础情形，$s \to^i_{\mathcal{R}} t$ 由 $\ell\gamma \to r\gamma$，且 $\ell\gamma$的所有子项都处于范式。因此，由于根据假设 $[\![\ell]\!]^{\mathcal{J}}_\alpha \succ [\![r]\!]^{\mathcal{J}}_\alpha$，引理5表明 $[\![\ell\gamma]\!]^{\mathcal{J}}_\alpha \succ [\![r\gamma]\!]^{\mathcal{J}}_\alpha$。

在归纳步骤中，我们结合引理6和归纳假设（IH）如下进行证明。假设 $\mathsf{s} \to^i_{\mathcal{R}} \mathsf{t}$，其中 $\mathsf{s} = \mathsf{s}'\mathsf{u} \to^i_{\mathcal{R}} \mathsf{s}''\mathsf{u}$，且 $\mathsf{s}' \to^i_{\mathcal{R}} \mathsf{s}''$。因此，$[\![\mathsf{s}'\mathsf{u}]\!]^{\mathcal{J}}_\alpha = [\![\mathsf{s}']\!]^{\mathcal{J}}_\alpha \cdot [\![\mathsf{u}]\!]^{\mathcal{J}}_\alpha = \mathrm{App}([\![\mathsf{s}']\!]^{\mathcal{J}}_\alpha, [\![\mathsf{u}]\!]^{\mathcal{J}}_\alpha)$；于是，归纳假设给出 $[\![\mathsf{s}']\!]^{\mathcal{J}}_\alpha \succ [\![\mathsf{s}'']\!]^{\mathcal{J}}_\alpha$，结合引理6可推出 $[\![\mathsf{s}]\!]^{\mathcal{J}}_\alpha = \mathrm{App}([\![\mathsf{s}']\!]^{\mathcal{J}}_\alpha, [\![\mathsf{u}]\!]^{\mathcal{J}}_\alpha) \succ \mathrm{App}([\![\mathsf{s}'']\!]^{\mathcal{J}}_\alpha, [\![\mathsf{u}]\!]^{\mathcal{J}}_\alpha) = [\![\mathsf{t}]\!]^{\mathcal{J}}_\alpha$。当 $\mathsf{s} \to^i_{\mathcal{R}} \mathsf{t}$，且 $\mathsf{s} = \mathsf{s}'\mathsf{u} \to^i_{\mathcal{R}} \mathsf{s}'\mathsf{u}'$ 时，证明类似。■

例3 令 $\mathsf{0}, \mathsf{1} : \iota, \mathsf{g} : \iota \Rightarrow \iota \Rightarrow \iota$，且 $\mathsf{f} : \iota \Rightarrow \iota \Rightarrow \iota \Rightarrow \iota$。Toyama [23]引入并由 $\mathcal{R} = \{\mathsf{g}\ x\ y \to x,\ \mathsf{g}\ x\ y \to y,\ \mathsf{f}\ \mathsf{0}\ \mathsf{1}\ z \to \mathsf{f}\ z\ z\ z\}$ 定义的重写系统，旨在说明项重写系统的不相交并的终止性并非模性质。事实上，它存在无限重写序列 $\mathsf{f}\ \mathsf{0}\ \mathsf{1}\ (\mathsf{g}\ \mathsf{0}\ \mathsf{1}) \to_{\mathcal{R}} \mathsf{f}\ (\mathsf{g}\ \mathsf{0}\ \mathsf{1})\ (\mathsf{g}\ \mathsf{0}\ \mathsf{1})\ (\mathsf{g}\ \mathsf{0}\ \mathsf{1}) \to^+_{\mathcal{R}} \mathsf{f}\ \mathsf{0}\ \mathsf{1}\ (\mathsf{g}\ \mathsf{0}\ \mathsf{1})$。然而，最内关系 $\to^i_{\mathcal{R}}$ 是终止的。为证明这一点，我们引入非数值大小概念。令 $\mathcal{J}_{\mathcal{B}}(\iota) = \mathcal{P}(T(\mathcal{F}, \mathcal{X}))$，即 $T(\mathcal{F}, \mathcal{X})$ 的所有子集构成的集合。这个集合按集合包含关系构成偏序，因此 $x \sqsupseteq y$ 当且仅当 $x \supseteq y$，这是一种拟序。考虑以下解释：

$$\mathcal{J}_0 = \langle 0, \{0\}\rangle \qquad \mathcal{J}_1 = \langle 0, \{1\}\rangle \qquad \mathcal{J}_{\mathsf{g}} = \langle (\boldsymbol{\lambda} xy.1), \boldsymbol{\lambda} xy.x \cup y\rangle \qquad \mathcal{J}_{\mathsf{f}} = \langle (\boldsymbol{\lambda} xyz.H(x,y)), \boldsymbol{\lambda} xyz.\emptyset\rangle,$$

其中，H 是由 $\mathsf{H(x, y)} = \mathsf{if\ x} \sqsupseteq \{0\} \wedge \mathsf{y} \sqsupseteq \{1\}\ \mathsf{then\ 1\ else\ 0}$ 定义的辅助函数。注意，H 是弱单调的，且范式中的所有项都被解释为大小为 ≤ 1 的集合。验证相容性很简单：$[\![\mathsf{g\ x\ y}]\!] = \langle 1, \mathsf{x} \cup \mathsf{y}\rangle \succ \langle 0, \mathsf{x}\rangle = [\![\mathsf{x}]\!]$ 和 $[\![\mathsf{g\ x\ y}]\!] = \langle 1, \mathsf{x} \cup \mathsf{y}\rangle \succ \langle 0, \mathsf{y}\rangle = [\![\mathsf{y}]\!]$；以及 $[\![\mathsf{f\ 0\ 1\ z}]\!] = \langle 1 \emptyset\rangle \succ \langle 0 \emptyset\rangle = [\![\mathsf{f\ z\ z\ z}]\!]$，因为 z 的任何实例化必然处于范式，因此不可能同时包含 0 和 1。

尽管这个例子并不真实，但从终止性的角度来看颇有意思。它表明，元组解释可以用于处理只能通过最内层策略终止的重写系统。

4 最内层运行时复杂度的多项式界

本节研究元组解释在兼容项重写系统的复杂度分析中的应用，即可在元组代数中得到解释的重写系统（$(\!|\cdot|\!)$，$\mathcal{J}$）。尽管在我们的框架中代价与大小是分开的，但它们是相互交织的概念（本节将对此作出精确定义），共同构成了我们直观上所说的项重写系统的“复杂度”。

4.1 可加的元组解释

为了得到 $\mathrm{irc}_{\mathcal{R}}(n)$ 的上界，只需给出所有满足 s（其中 $|s| \leq n$）的项的代价分量 $[\![s]\!]^{\mathsf{c}}$ 的上界。此外，由于基本项的形式为 $\mathsf{f}\ d_1 \ldots d_m$，数据项的大小在我们的分析中起着重要作用。下文解释类型时，我们采用解释键的默认选择；即 $\mathcal{J}_{\mathcal{B}}(\iota) = \mathbb{N}^{K[\iota]}$，其中每个 $\iota \in \mathcal{B}$ 均对应 $K[\iota] \geq 1$。

给定 $\sigma = \iota_1 \Rightarrow \ldots \Rightarrow \iota_m \Rightarrow \kappa$，$(\!|\sigma|\!)$ 的大小分量是 $\mathcal{S}_\sigma = \mathbb{N}^{K[\iota_1]} \Longrightarrow \ldots \Longrightarrow \mathbb{N}^{K[\iota_m]} \Longrightarrow \mathbb{N}^{K[\kappa]}$。大小函数 $f^{\mathsf{s}} \in \mathcal{S}_\sigma$ 完全应用后，可以用函数分量表示。因此，$f_{\mathsf{s}}(x1, \ldots, x_m) = \left\langle f_{\mathsf{s}}\mathsf{1}(x1, \ldots, x_m) \ldots, f_{\mathsf{s}K[\kappa]}(x1, \ldots, x_m)\right\rangle$。

定义 10 设 σ是一个类型，且 $\mathsf{f\ s} \in \mathcal{S}_\sigma$。若其每个分量函数 $\mathsf{f\ s1}, \ldots, \mathsf{f\ sK}_{[\kappa]}$ 都被一个正线性多项式所上界，则称大小函数 f s 是线性有界的；也就是说，存在一个正常数 $\mathsf{a} \in \mathbb{N}$，使得对于所有 $1 \leq \mathsf{l} \leq \mathsf{m}$，$\mathsf{f\ sl\ (x1, \ldots, x_m)} \leq \mathsf{a}(1 + \sum_{i=1}^m \sum_{=1}^{K[\iota_i]} \mathsf{j\ x}_{ij})$。类似地，若存在一个常数 $\mathsf{a} \in \mathbb{N}$，使得 $\sum^{K[\kappa]} \mathsf{l}_{=1} \mathsf{f\ sl\ (x1, \ldots, x_m)} \leq \mathsf{a} + \sum_{i=1}^m \sum_{=1}^{K[\iota_i]} \mathsf{j\ x}_{ij}$，则称 f s 是可加的。

请注意，根据此定义，线性有界（或可加的）大小函数并不要求是线性的（或可加的），而只要求被一个线性（或可加的）函数所上界。因此，例如可以使用 $\min(x,\ 2y)$，但不能使用 xy。大小解释不一定能界定数据项的绝对大小。例如，我们可以将数据构造器 $\mathsf{c} : \iota \Rightarrow \kappa$ 解释为 $\mathcal{J}_{\mathsf{sc}} = \boldsymbol{\lambda} x.\lfloor x/2 \rfloor$，从而得到 $|d| \geq [\![d]\!]^s$。这在处理次线性解释时尤其有用。

下一个引理保证，通过对构造子采用可加的解释，数据项的大小解释与其绝对大小成正比：

引理 7 设 $\mathcal{R}$ 是一个与成本–大小元组代数 $(\Vdash\cdot\Vdash, \mathcal{J})$ 相容的项重写系统。

(i) 假设对所有数据构造器 c，$\mathcal{J}^s_{\mathsf{c}}$ 都是可加的，那么对所有数据项 d：若 $|d| \leq n$，则存在常数 $b > 0$ 使得 $[\![d]\!]^{\mathsf{s}}_l \leq bn$；对于 $[\![d]\!]$ 的每个大小分量 $[\![d]\!]^{\mathsf{s}}_l$，均如此。*(ii)* 假设对所有数据构造器 c，$\mathcal{J}^s_{\mathsf{c}}$ 都是线性有界的，那么对所有数据项 d：若 $|d| \leq n$，则存在常数 $b > 0$ 使得 $[\![d]\!]^{\mathsf{s}}_l \leq 2^{bn}$；对于 $[\![d]\!]$ 的每个大小分量 $[\![d]\!]^{\mathsf{s}}_l$，均如此。

(ii) 中的界是紧的。事实上，定义（解释 $\mathcal{R}_{\mathsf{add}}$ 时）：$\mathcal{J}_0 = \langle 0, 1\rangle, \mathcal{J}_{\mathsf{s}} = \langle (\boldsymbol{\lambda} x.0)\ \boldsymbol{\lambda} x.2x+1\rangle$，以及 $\mathcal{J}_{\mathsf{add}} = \langle (\boldsymbol{\lambda} xy.y+1)\ \boldsymbol{\lambda} xy.x+y\rangle$。在这种情况下，对于数据项 $\mathrm{n} = \mathrm{s}^n(0)$，其大小解释恰好为 $[\![\mathrm{n}]\!]^{\mathsf{s}} = 2^n + n \leq 2^{|\mathrm{n}|}$。然而，尽管这一选择与 $\mathcal{R}_{\mathsf{add}}$ 兼容，从而证明了其终止性，但它会给 $\mathrm{irc}^{\mathcal{R}_{\mathsf{add}}}$ 带来指数级开销，而该项是线性有界的（见例 4）。在复杂度分析中，如此巨大的高估并不可取。这种现象为数据构造器的解释设定了严格的上界：也就是说，我们希望构造子的大小解释是可加的。容易证明，例 2 中自然数和列表的大小分量是可加的。

定义 11 如果对于每个 $\mathsf{c} \in \mathcal{C}$，$\mathcal{J}^{\mathsf{s}}_{\mathsf{c}}$ 都是可加的，我们就称解释 $\mathcal{J}$ 是可加的 .

4.2 成本有界元组解释

下文中，我们考虑具有加法解释的重写系统。

定义 12 设 σ 是一个类型，且 $fff_{\mathsf{c}} \in \mathcal{C}_\sigma$。如果每个 e_i、$0 \leq i \leq m$ 都是线性有界（可加有界）的，我们就称 fff_{c}（如形式 (1) 所示）是线性有界（可加有界）的。此外，如果 $_{\mathsf{cf}}\mathcal{J}$ 和 $_{\mathsf{sf}}\mathcal{J}$ 都由 f 有界，则 $\mathcal{J}_{\mathsf{f}}$ 由函数 f 有界。

在下一个引理中，我们汇总了在能够为解释的成本-规模分量提供界的情况下，所导出的内在运行复杂度的适当上界。

引理 8 假设 $\mathcal{R}$ 是一个与元组代数 $(\Vdash\cdot\Vdash、\ \mathcal{J})$ 相容的项重写系统，则：

(i) 如果对于所有 $\mathsf{f} \in \mathcal{F}$, $\mathcal{J}_{\mathsf{sf}}$ 是对数级的，且 $\mathcal{J}_{\mathsf{cf}}$ 是加法有界的，那么 $\mathtt{irc}_{\mathcal{R}}(n) \in \mathcal{O}(\log n)$；*(ii)* 如果对于所有 $\mathsf{f} \in \mathcal{F}$, $\mathcal{J}_{\mathsf{f}}$ 是加法有界的，那么 $\mathtt{irc}_{\mathcal{R}}(n) \in \mathcal{O}(n)$；以及 *(iii)* 如果对于所有定义符号 f 和构造子 c，$\mathcal{J}_{\mathsf{c}}$ 是加法有界的，且 $\mathcal{J}_{\mathsf{f}}$ 有多项式上界

d,

那么对于某个 $k \in \mathbb{N}$，$\mathtt{irc}_{\mathcal{R}}(n) \in \mathcal{O}(n^k)$。

例 4 我们通过解释例 1 中的函数来说明这一行为。例 2 给出了构造子的解释。

$$
\begin{array}{llll}
\mathcal{J}_{\mathsf{add}} & = & \langle (\boldsymbol{\lambda} xy.y+1), \boldsymbol{\lambda} xy.x+y\rangle & \mathcal{J}_{\mathsf{sum}} = \langle (\boldsymbol{\lambda} q.2q_{\mathsf{l}} + q_{\mathsf{l}} q_{\mathsf{m}}), \boldsymbol{\lambda} q.q_{\mathsf{l}} q_{\mathsf{m}}\rangle \\
\mathcal{J}_{\mathsf{minus}} & = & \langle (\boldsymbol{\lambda} xy.y+1), \boldsymbol{\lambda} xy.x\rangle & \mathcal{J}_{\mathtt{rev}} = \left\langle (\boldsymbol{\lambda} q.q_{\mathsf{l}} + \tfrac{q_{\mathsf{l}}(q_{\mathsf{l}}+1)}{2} + 1), \boldsymbol{\lambda} q.q \right\rangle \\
\mathcal{J}_{\mathsf{quot}} & = & \langle (\boldsymbol{\lambda} xy.x+xy+1), \boldsymbol{\lambda} xy.x\rangle & \\
\mathcal{J}_{\mathsf{append}} & = & \langle (\boldsymbol{\lambda} ql.q_{\mathsf{l}}+1), \boldsymbol{\lambda} ql.\langle q_{\mathsf{l}} + l_{\mathsf{l}}, \max(q_{\mathsf{m}}, q_{\mathsf{m}})\rangle\rangle &
\end{array}
$$

检查这一解释的相容性很简单。请注意，在例 1 中定义函数 f 的每组规则里，大小分量都是可加的，而代价分量都是多项式有界的。根据引理 8 的情形 (b)，我们有 $\mathrm{irc}_{\mathcal{R}_{\mathsf{add}}}$、$\mathrm{irc}_{\mathcal{R}_{\mathsf{append}}}$ 和 $\mathrm{irc}_{\mathcal{R}_{\mathsf{minus}}}$ 是线性的。对于 $\mathrm{irc}_{\mathcal{R}_{\mathsf{quot}}}$、$\mathrm{irc}_{\mathcal{R}_{\mathsf{sum}}}$ 和 $\mathrm{irc}_{\mathcal{R}_{\mathsf{rev}}}$，可以推导出二次上界。

回顾一下大小分量所具有的语义含义（见例 2），可以发现，解释的代价分量不仅能界定 $\mathcal{R}_{\mathsf{f}}$ 的最内层运行时复杂度，还能提供额外信息，说明各个大小分量在重写代价中所起的作用。例如：两个数相加的代价仅取决于加法的第二个参数的大小；对列表中的每个元素求和，其代价与列表长度呈线性关系，与列表长度和最大元素呈非线性关系。这在程序分析中特别有用，因为通过分析解释本身的形态，可以检测出潜在的高代价操作。

5 自动化

本节概述一种寻找成本–大小元组解释的流程。我们的目标是找到能够保证相关重写系统的运行时复杂度具有多项式界的解释。因此，我们有以下条件：(i) 所选的解释键针对 $\mathbb{N}$；(ii) 构造子的大小解释是加法有界的；(iii) 函数符号的解释是多项式有界的。

参数化解释。回想一下，在论文前文中，我们为大小分量赋予了直观含义。在完全自动化的设置中，不允许人工指导，所有基类型 ι 都从$K[\iota] = 1$ 开始，递增至预先设定的上界 K。这个最大上界 K 是限制搜索空间并确保搜索过程终止所必需的。

大致而言，该过程如下运行。数据构造器的解释设定为可加的。因此，如果 c : $\iota_1 \Rightarrow \ldots \Rightarrow \iota_m \Rightarrow \kappa$是一个数据构造器，其大小解释为 $\boldsymbol{\lambda} x_1 \ldots x_m . a + \sum_{i=1}^{m} \sum_{=1}^{K[\iota_i]}{}_j x^i{}_j$，其中$a$ 是由搜索过程确定的参数。我们称这种解释形式由系数a参数化。下一步是为定义符号 f $\in \mathcal{F}$选择（参数化）解释。与构造子不同，构造子的代价分量是零值函数，大小分量是可加的；而对于定义符号 f $\in \mathcal{F}$的代价分量和大小分量，我们可以选择任何多项式有界的函数。

然而，我们可以用来选择定义符号解释的函数类过于庞大。因此，我们将搜索空间限制在一类有限的多项式有界函数中：最大值多项式，即由多项式项和最大值函数组合而成的函数。例如，例 2 中 cons 的解释和例 4 中 append 的解释都是最大值多项式。随后，我们为代价和大小分量选择由其系数参数化的通用最大值多项式。回想一下，我们希望找到满足相容性条件的解释，即，对于任意 α，$[\![\ell]\!]_\alpha \succ [\![r]\!]_\alpha$。因此，如果我们选取由其系数参数化的最大值多项式，就会得到一组约束，必须求解这些约束才能确定有效的解释。

示例 5 让我们用一个简单的系统来说明上述思想，该系统定义了自然数上的函数 dbl。因此，我们考虑由以下规则组成的系统：dbl $0 \to 0$ 和 dbl (s x) $\to$ s (s (dbl x))。我们选择以下参数化解释

$$\mathcal{J}_0 = \langle 0, a_0 \rangle \qquad \mathcal{J}_{\mathsf{s}} = \langle (\boldsymbol{\lambda} x.0), \boldsymbol{\lambda} x.x + b_0 \rangle \qquad \mathcal{J}_{\mathtt{dbl}} = \langle (\boldsymbol{\lambda} x.c_1 x + c_0), \boldsymbol{\lambda} x.d_1 x + d_0 \rangle ,$$

它满足上述条件 (i)-(iii)。上述解释是参数化的，也就是说，系数$a_0, b_0, c_0, c_1, d_0, d_1$尚待确定。第一条规则的相容性条件给出：

$$[\![\texttt{dbl 0}]\!] \succ [\![\texttt{0}]\!] \implies \langle (c_1a_0 + c_0), d_1a_0 + d_0\rangle \succ \langle 0, a_0\rangle,$$

因此需要以下公式成立：$C_0 = (c_1a_0 + c_0 > 0) \wedge (d_1a_0 + d_0 \geq a_0)$。另一方面，第二条规则的相容性条件给出如下内容：

$$[\![\texttt{dbl}\ (\texttt{s}\ x)]\!] \succ [\![\texttt{s}\ (\texttt{s}\ (\texttt{dbl}\ x))]\!] \implies \langle (c_1x + c_1b_0 + c_0), d_1x + d_1b_0 + d_0\rangle \succ \langle (c_1x + c_0), d_1x + d_0 + 2b_0\rangle,$$

因此需要公式成立

$$C_1 = (c_1x + c_1b_0 + c_0 > c_1x + c_0) \wedge (d_1x + d_1b_0 + d_0 \geq d_1x + d_0 + 2b_0).$$

因此，我们要为约束C_0,C_1 寻找见证，范围为 $\mathbb{N}$。为此，我们可以使用 SMT 求解器。

上面的示例本身非常简单，但运用了我们过程的主要思想。本质上，我们为 $\mathcal{F}$ 中的函数符号选择参数化解释，并求解由相容性条件产生的约束。正如我们在例 4 中所见，成本–大小解释可能变得复杂，因此搜索过程中需要更多解释形式。我们将在下文描述这样一个过程。它具有模块化特点，因为它由选择器策略 $\mathcal{S}$ 和约束求解器参数化。选择器策略是一种算法，用于为 $\mathcal{F}$ 中的每个函数符号选择参数化解释。例如，在上面的示例中，我们为所有函数符号都选择了线性参数化形式 。

主过程

参数： 选择器算法 $\mathcal{S}$ 以及用于非线性整数算术的约束求解器。**数据输入：** 一个基于语法签名 $(\mathcal{B}, \mathcal{F}, \mathrm{ar})$ 的项重写系统 $\mathcal{R}$。

输出： 如果能找到满足相容性的成本–大小元组解释，则返回 YES；如果执行了以下所有步骤但未能找到解释，则返回 MAYBE[1]。

1. 将 $\mathcal{F}$ 划分为两个互不相交的集合，分别包含构造子和定义符号，即 $\mathcal{F} = \mathcal{C} \uplus \mathcal{D}$。

2. 对于每个构造子 $\mathsf{c} : \iota_1 \Rightarrow \ldots \Rightarrow \iota_m \Rightarrow \kappa$，选择其代价解释为取零值的成本函数；大小解释为可加的。

3. 将 $\mathcal{D}$ 划分为集合 $\mathcal{D}_1, \ldots, \mathcal{D}_n$，使得对于每个 $\mathsf{f} \in \mathcal{D}_i$，其中 $1 \leq i \leq n$，定义 f 的规则中出现的所有函数符号要么是构造子，要么属于 $\mathcal{D}_1 \cup \cdots \cup \mathcal{D}_i$。

4. 对于每个 $1 \leq i \leq n$，根据选择器策略 $\mathcal{S}$（将在下文定义），为 $\mathcal{D}_i$ 中的符号选择一种解释形式 。

   • 在 $\mathcal{S}$ 上标记所选的解释形式，以免这一步失败时再次选择相同的形式。

   • 如果无法通过 $\mathcal{S}$ 做出选择，就停止并返回可能。

5. 如果 $\mathsf{f}\ \ell_1\ \ldots\ \ell_k \to r$ 是一个类型为 ι、且包含 $\mathsf{f} \in \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_i$ 的规则，简化 $[\![\mathsf{f}\ \ell_1\ \ldots\ \ell_k]\!] \succ [\![r]\!]$，使结果成为一组不依赖任何已解释变量的不等式约束$C$（我们将在下文定义这一步简化操作）。

   • 如果这 如果简化步骤失败，我们就返回步骤 4，选择另一种解释形式。

[1]请注意，在我们的设定中，不可能返回否。

6. 检查是否C 成立。
   - 如果C 中的所有约束都成立且$i < n$，这意味着我们可以定向所有以函数符号开头的规则 $\mathcal{D}_i$，因此带着$i := i + 1$进入第 4 步。
   - 如果C 中的所有约束都成立且$i = n$，那么我们可以定向所有规则 $\mathcal{R}$，停止并返回是。
   - 否则，将$K[\iota]$加一，更新构造子的加法大小解释，并返回第 4 步，选择另一种解释形式。

上述过程还有两个关键方面有待定义：$\mathcal{S}$ 选择解释形式的策略，以及第 6 步中的约束求解器。

基于策略的元组解释搜索。</style}直观地说，选择器策略 是一种算法，用于为 中的定义符号选择参数化解释。例如，我们可以从列表中随机挑选一种解释形式（“盲选”策略）； 也可以从一系列可能的尝试中逐步选择解释（**“渐进式”**策略）；或者根据语法模式选择解释（**“模式”**策略）。

下面的定义列出了我们考虑的一些解释形式。它们基于 [9,22] 中研究的类。参数化解释是通过考察定义符号的类型构造的。

定义 13（解释形式） 令 $\sigma = \iota_1 \Rightarrow \ldots \Rightarrow \iota_m \Rightarrow \kappa$，并且令下列形式中出现的每个$f_{ij}$都是定义在 $\mathcal{S}_\sigma$ 上的加法有界弱单调函数。对于$f(\vec{x})$，我们表示f对每个参数$x_1, \ldots, x_m$的应用。

- 加法类包含具有以下形式的加法有界代价—大小泛函：

$$\boldsymbol{\lambda} x_1 \ldots x_m. \sum_{i=1}^{m} \sum_{j=1}^{K[\iota_i]} x_{ij} + b_0 + f(\vec{x})$$

- 线性类包含写作如下形式的代价—大小泛函：

$$\boldsymbol{\lambda} x_1 \ldots x_m. \sum_{i=1}^{m} \sum_{j=1}^{K[\iota_i]} a_{ij} x_{ij} f_{ij}(\vec{x})$$

- 简单类包含写作如下形式的代价—大小泛函：

$$\boldsymbol{\lambda} x_1 \ldots x_m. \sum_{i=1}^{m} \sum_{j=1}^{K[\iota_i]} a_{ij} x_{ij}^{k_{ij}} f_{ij}(\vec{x}), \text{ such that each } k_{ij} \in \{0,1\}$$

- 最后，二次类包含这样的代价—大小泛函：其中允许变量构成一般乘积，次数最高为 2：

$$\boldsymbol{\lambda} x_1 \ldots x_m. \sum_{i=1}^{m} \sum_{j=1}^{K[\iota_i]} a_{ij} x_{ij}^{k_{ij}} f_{ij}(\vec{x}), \text{ such that each } k_{ij} \in \{0,1,2\}$$

- 简单二次类包含由一个简单泛函与一个二次部分求和构成的代价—大小泛函：

$$\boldsymbol{\lambda} x_1 \ldots x_m. \sum_{i=1}^{m} \sum_{j=1}^{K[\iota_i]} a_{ij} x_{ij}^{k_{ij}} f_{ij}(\vec{x}) + \sum_{i=1}^{m} \sum_{j=1}^{K[\iota_i]} a_{ij} x_{ij}^{l_{ij}} f_{ij}(\vec{x}),$$

  与 $k_{ij} \in \{0,1\}$ 和 $l_{ij} \in \{0,1,2\}$。

因此，盲目策略会从上述形式中随机选择一种。增量策略则按顺序选择解释，从可加的解释到二次解释。模式策略的实现稍微复杂一些，因为我们需要根据规则的形式进行启发式分析。例如，所有形如 $\mathrm{f}\,x_1 \ldots x_m \to x_i$ 的规则都具有常数成本函数（$\boldsymbol{\lambda} x_1 \ldots x_m.1$）和可加的大小分量。像模式 $C[x] \to D[x,x]$，这样重复变量的规则，会使代价至少具有二次上界。请注意，本文中的所有二次复杂度都属于这种情况。选择器算法的具体实现决定了主过程中查找解释的效率。

为了简化约束 $[\![\ell]\!] \succ [\![\mathrm{r}]\!]$，我们必须简化多项式（最大值多项式）之间的不等式。为了简化多项式（最大值多项式）的形式，我们需要比较多项式 $\mathrm{P}^{\mathrm{c}}_{\ell} > \mathrm{R}^{\mathrm{c}}_{r}$ 和 $\mathrm{P}^{\mathrm{s}}_{\ell_1} \sqsupseteq \mathrm{P}^{\mathrm{s}}_{r_1} \wedge \cdots \wedge \mathrm{P}^{\mathrm{s}}_{\ell_{K[\tau]}} \sqsupseteq \mathrm{P}^{\mathrm{s}}_{r_{K[\kappa]}}$。随后，这些条件会被转化为量词自由非线性整数算术（QF NIA）公式，并提交给 SMT 求解器，见 [10]。最大值多项式使用规则 $\mathrm{max(x,\,y)} + \mathrm{z} \rightsquigarrow \mathrm{max(x} + \mathrm{z,y} + \mathrm{z)}$ 和 $\mathrm{max(x,y)z} \rightsquigarrow \mathrm{max(xz,yz)}$ 进行化简。结果的形式为 $\mathrm{max}_i \mathrm{P}_i$，其中每个 P_i 都是不含 max 的多项式 [7]。

6 结论

本文展示了如何调整成本–大小元组对，使其能够处理最内层重写。类型感知的代数解释风格提供了必要的工具，用于处理最内层终止性，并建立兼容项重写系统最内层运行时复杂度的上界。我们提出了适用于兼容系统的 $\mathrm{irc}_{\mathcal{R}}$ 可行（多项式）上界的充分条件，这与文献中的相关研究一致。这一研究方向远未结束。由于寻找解释可能十分繁琐，我们当前的后续工作是开发新的策略和解释形式。例如，我们希望将解释类别扩展到最大多项式之外，例如对数泛函。这有望大幅提升我们工具的效率。

致谢。 我们谨向 Cynthia Kop 致谢，感谢她在本文撰写过程中提供的宝贵讨论和指导；感谢 Niels van der Weide、Marcos Bueno 和 Edna Gomes 仔细校阅本文的多个稿本；并感谢匿名评审人提出宝贵意见，帮助我们改进本文。

参考文献

[1] M. Avanzini & G. Moser（2008）：基于重写的复杂度分析。载于：*FLOPS* 论文集，第 130–146 页，doi:10.1007/978-3-540-78969-7 11。[2] P. Baillot & U. Dal Lago（2016）：高阶解释与程序复杂度。《IC》，第 56–81 页，doi:10.1016/j.ic.2015.12.008。[3] A. Ben Cherifa & P. Lescanne（1987）：通过多项式解释实现重写系统的终止性及其实现。《计算机编程科学》9(2)，第 137–159 页，doi:10.1016/0167-6423(87)90030-X。[4] G. Bonfante, A. Cichon, J.-Y. Marion & H. Touzet（2001）：利用多项式解释终止性证明的算法。《函数式编程杂志》11(1)，第 33–53 页，doi:10.1017/S0956796800003877。[5] G. Bonfante, J. Marion & J. Moyen（2001）：关于带空间界认证的字典序终止排序。载于：*PSI* 论文集，第 482–493 页，doi:10.1007/3-540-45575-2 46。[6] A. Cichon & P. Lescanne（1992）：多项式解释与算法复杂度。载于：*_CADE*，第 139–147 页，doi:10.1007/3-540-55602-8 161。

[7] M. Codish, I. Gonopolskiy, A. M. Ben-Amram, C. Fuhs & J. Giesl（2011）：使用整数上的单调性约束进行基于 SAT 的终止性分析。《Theory and Practice of Logic Programming》11(4-5)，第 503–520 页，doi:10.1017/S1471068411000147。[8] E. Contejan, C. March´e, A. P. Tom´as & X. Urbain（2005）：使用多项式解释机械化证明终止性。《JAR》34(34)，doi:10.1007/s10817-005-9022-x。[9] Evelyne Contejean, Claude March´e, Ana Paula Tom´as & Xavier Urbain（2005）：使用多项式解释机械化证明终止性。《JAR》34(4)，第 325–363 页，doi:10.1007/s10817-005-9022-x。[10] J. Giesl, C. Aschermann, M. Brockschmidt, F. Emmes, F. Frohn, C. Fuhs, J. Hensel, C. Otto, M. Plucker, P. Schneider-Kamp, T. Stroder, S. Swiderski & R. Thiemann（2017）：使用 AProVE 自动分析程序终止性与复杂度。《JAR》58，第 3–31 页，doi:10.1007/s10817-016-9388-y。[11] N. Hirokawa & G. Moser（2008）：基于依赖对方法的自动化复杂度分析。载于：IJCAR 论文集，第 364–379 页，doi:10.1007/978-3-540-71070-7 32。[12] D. Hofbauer（1992）：多重集路径序的终止性证明蕴含原始递归推导长度。TCS 论文集，doi:10.1007/3-540-53162-9 50。[13] D. Hofbauer（2001）：通过上下文相关解释进行终止性证明。载于：RTA 论文集，第 108–121 页，doi:10.1007/3-540-45127-7 10。[14] D. Hofbauer & C. Lautemann（1989）：终止性证明与推导长度。载于：RTA 论文集，第 167–177 页，doi:10.1007/3-540-51081-8 107。[15] G. Huet & D.C Oppen（1980）：方程与重写规则：综述。《形式语言理论：观点与开放问题》，Loria。可访问：http://rewriting.loria.fr/documents/CS-TR-80-785.pdf。[16] C. Kop & D. Vale（2021）：用于高阶复杂度的元组解释。载于：FSCD，第 31:1–31:22 页，doi:10.4230/LIPIcs.FSCD.2021.31。[17] C. Lautemann（1988）：关于多项式解释的一则短评。《EATCS 公报》第 36 卷，第 129–131 页。[18] F. Mitterwallner & A. Middeldorp（2022）：在 ℕ 上的多项式终止性不可判定。载于：FSCD 论文集，第 27:1–27:17 页，doi:10.4230/LIPIcs.FSCD.2022.27。[19] G. Moser（2017）：通过重写进行统一资源分析：优势与不足（特邀报告）。载于：FSCD 论文集，第 2:1–2:10 页，doi:10.4230/LIPIcs.FSCD.2017.2。[20] G. Moser, A. Schnabl & J. Waldmann（2008）：基于矩阵与上下文相关解释的项重写复杂度分析。载于：IARCS 论文集，第 304–315 页，doi:10.4230/LIPIcs.FSTTCS.2008.1762。[21] L. Noschinski, F. Emmes & J. Giesl（2011）：用于项重写系统最内复杂度分析的依赖对框架。载于：CADE-23，第 422–438 页，doi:10.1007/978-3-642-22438-6 32。[22] J. Steinbach（1992）：证明多项式为正。载于：FSTTCS 92 论文集，柏林，海德堡，第 191–202 页，doi:10.1007/3-540-56287-7 105。[23] Yoshihito T.（1987）：项重写系统直和的终止性反例。《Information Processing Letters》25(3)，第 141–143 页，doi:10.1016/0020-0190(87)90122-0。[24] A. Weiermann（1995）：通过字典序路径序对项重写系统进行终止性证明，蕴含多重递归推导长度。《TCS》，doi:10.1016/0304-3975(94)00135-6。[25] A. Yamada（2022）：用于项重写终止性的元组解释。《自动推理杂志》，doi:10.1007s10817-022-09640-4。