arXiv:2302.03074v1 [cs.LO] 6 Feb 2023

通过递归方案的有理函数

SIDDHARTH BHASKAR、JANE CHANDLEE 和 ADAM JARDINE

1. Introduction

形式语言理论的核心概念之一是正则性。有限字母表上的有限字符串所构成的正则语言这一典型示例，已沿多个方向得到推广。如今，针对其他类型的数据（例如无限字符串和树），以及针对函数而非语言，都有了稳健的正则性概念。每种正则性概念通常都可以从三个角度加以刻画：机械地（通过有限自动机）、代数地（通过文法）和逻辑地（通过单子二阶逻辑的片段）。形式语言理论的大量研究工作都致力于拓展正则性的概念，并使其更加稳健。

本文也不例外。我们关注的是有限字符串上的函数类。通过自动机理论或许最容易理解这些函数：正则函数是指任何可由变换器计算的函数；变换器是一种有限自动机，其转移可以向输出端打印字符串。然而，情况很快就变得复杂起来：对于变换器而言，确定性、非确定性、单向和双向自动机之间的等价关系分裂为三类：

(1) 单向确定性转换器计算子序列函数这一类。[1] 根据转换器读取字符串的方向，这些函数还可进一步分为左子序列函数和右子序列函数。(2) 单向非确定性转换器计算严格更大的有理函数这一类；尽管转换器是单向的，这些函数在字符串反转下仍然封闭，而且无需再分为左类和右类。[2](3) 双向确定性和非确定性转换器计算更严格更大的正则函数这一类。（显然，这些函数在字符串反转下封闭。）

这些类也都具有逻辑刻画。这些刻画体现了有限模型论的思想：我们将固定字母表 Σ 上的每个字符串解释为固定签名上的有限结构，固定签名也称为 Σ。该结构的论域是字符串的索引集合；对于字母表中的每个字符，我们都有一个谓词，用来选出包含该字符的索引；此外，我们还可以用某种方式比较或操作索引，例如采用线性序，或后继函数和前驱函数。

编辑部收到日期：2023年2月8日。[1]有时称为 *sequential*，参见 [Sak09]。[2]请注意，在讨论非确定性转换器时，我们仅考虑确实计算函数的转换器，即给定输入的输出不随非确定性选择序列而改变。

预印本，已提交至《计算机科学中的逻辑方法》

© 通过递归方案的有理函数 CC BY 知识共享

在这种将字符串与有限结构对应起来的做法下，字符串语言可以对应为谱。这是自动机理论与复杂性类的逻辑刻画之间的基本桥梁。典型的此类刻画是将正则字符串语言与单子二阶（MSO）逻辑 [Bö60] 中语句的谱对应起来。Engelfriet 和 Hoogeboom 将这一结果扩展到了正则函数和 MSO可定义解释 [EH01]。

关于解释的一点说明。给定两个签名 L和 K，解释 $L \to K$ 为 L的每个符号指定一个K-公式。这就给出了从 K-结构到 L-结构的映射：给定一个 K-结构，我们可以用其 K定义来解释任意 L符号。[3] 如果 L 和 K 分别是字符串字母表 Σ 和 Γ，那么我们可以将有限的 L-结构集和 K-结构集分别与$\Sigma^\star$ 和 $\Gamma^\star$ 等同，而解释 $L \to K$ 定义了一个映射 $\Gamma^\star \to \Sigma^\star$。

在形式语言理论中，我们感兴趣的是定义这样的函数：它使字符串的长度至多增加一个常数乘法因子。遵循 Engelfriet 和 Hoogeboom [EH01]（最终可追溯至 Courcelle [Cou94]）的工作，我们将所定义L字符串的索引与原始 K字符串的单个索引对应起来。若要使常数因子超过 1，我们可以固定地重复这一操作有限次，并合并由此得到的副本。此后，我们所说的解释，均指这种意义下的解释。[4]

保序性。固定一个从 Σ 到 Γ 的解释 π。那么，如果 s 是某个 Γ 字符串，且 $|s|$ 是其索引集合，则 π 定义一个 Σ 字符串，其索引集合为某个固定 $m \in \omega$ 的 $m \times |s|$。存在一个 π 中的 Γ 公式，用于定义这些索引在输出字符串中的顺序。保序解释则不采用这种方式：并非如此，不存在这样的公式。相反，必须按照 $m \times |s|$ 上自然的字典序来考察输出字符串的索引。Bojanczyk [Boj14] 和 Filiot [Fil15] 发现，将 MSO解释细化为保序MSO解释后，所得的可计算函数类便从正则函数降为有理函数。

布尔单子递归方案。本文作者 [BCJO20] 提出了布尔单子递归方案（BMRS）的概念。这是一种针对字符串的弱编程语言；他们发现，单向保序BMRS解释恰好计算子序列函数。BMRS 的提出源于计算音系学方面的考量：它提供了一种形式体系，既能同时确保“正确”的计算上界，又足够灵活，使语言学上重要的现象能够轻易地从代码中读出 [CJ21]。

我们的贡献。句法复合问题是：给定两个计算函数 g 和 f 的解释，求一个计算函数 $g \circ f$ 的解释。本文解决了保序 BMRS 解释的句法复合问题。

标准解释天然具有复合性。给定解释 $\pi : L \to K$和 $\rho : K \to J$，我们可以通过在 π 的公式中，将每个 aK-符号的出现替换为 ρ中对应的 J-公式，从而构造解释 $L \to J$。

[3]注意从 K 到 L 的方向是反过来的；经过恰当的形式化，我们可以定义一个从解释到它们所定义映射的反变语义函子。[4]术语 *interpretation* 源自模型论，在模型论中，输出结构的论域通常被视为输入结构的 *tuples*，而非其副本。（例如，复数可以视为实数对。）形式语言理论中使用的另一个术语是 *transduction*。

然而，保序解释并不天然具有复合性：它们使用输入字符串上的索引顺序，却不在输出字符串上定义这些索引。更复杂的是，保序解释可以跳过输出中未定义任何字符的索引。[5] 直观来看，复合保序解释似乎需要一些基本算术运算来计算被跳过的索引，而 BMRS 不具备任何计数机制。[6]

我们的核心技术贡献是对保序解释的句法复合问题进行细致分析。我们将这一句法复合问题分解为四个部分，其中三个部分十分常规，几乎适用于任何逻辑。最后一个部分则很困难；在缺乏计数的情况下，它似乎恰恰需要 BMRS 的计算能力。

作为这项分析的结果，我们发现保序 BMRS 解释能够计算有理函数类。任何有理函数都可以分解为一个左、右子序列函数 [EM65] 的复合。根据我们之前的论文，这两者都可以通过单向 BMRS 计算；而根据本文，它们的复合可以通过 BMRS 计算。换言之，虽然很容易看出每个 BMRS 解释都是 MSO 解释，但我们也证明了反过来同样成立。这一结果结合我们之前论文的结果表明，子序列函数和有理函数都具有自然的 BMRS 刻画——而 MSO 逻辑目前尚无此类刻画。

相关工作。各种编程语言的一元片段从方案学兴起之初便已受到关注，因为人们常发现，这些片段兼具不俗的表达能力与比一般递归程序更易处理的特点。在形式语言理论中，MSO 等一元逻辑占据着至关重要的地位。MSO 与各种配备递归机制的一元一阶逻辑（例如一元最小不动点逻辑和一元 Datalog）之间的关系十分重要；已知这些逻辑在字符串和树数据上至少对于布尔查询而言，与 MSO 具有同等的表达能力 [Sch06].[7]

尽管我们的方法有所不同——例如，我们选择的字符串基本操作，以及考虑解释而非简单查询等——但我们推测，已知结果或许可以拼凑出我们关于保序 BMRS 解释刻画有理函数的结论。然而，我们认为真正新颖的是我们给出的证明：它源于解决保序 BMRS 解释的句法复合问题，而非直接模拟给定的 MSO 解释。

本文结构。第 2 节给出字符串的模型论定义，第 3 节定义在这些结构上的 BMRS，第 4 节定义保序 BMRS 解释，并说明它们如何定义字符串上的函数。第 5 节和第 6 节介绍技术性工作，展示如何复合保序 BMRS 解释；第 7 节运用这一结果证明，这类解释能够刻画有理函数。最后，第 8 节讨论进一步的研究方向。

[5]因此，即使输出的索引与 $m \times |s|$ 相对应，输出字符串的长度也可能小于 $m|s|$。[6]我们用 BMRS 表示布尔单子递归方案的单数和复数形式。通过逆构词法，这个缩写在复数形式中读作 $beamers$，在单数形式中读作 $beamer$。[7]至少有一篇论文指出，这类结果在数据库界早已是众所周知的；参见 [GK04]。

2. Strings as finite structures

我们关注有限字母表上的有限字符串，或者说，关注由某个非空符号集构成的有限字符串。通常用大写希腊字母（例如 Σ、Γ、Δ）表示字母表，用小写拉丁字母（例如 s、t）表示字符串。

字母表 Σ 上的字符串可以与一个一阶签名相关联，该签名也写作Σ。

定义1. 给定一个字母表 Σ，签名 Σ 由以下部分组成：
- 对每个字符 $\sigma \in \Sigma$，都有一个单子的（即一元）关系符号 σ，
- 单子的关系符号 max 和 min，以及
- 一元函数符号 S 和 P（分别表示后继和前驱）。

在需要区分时，我们将字母表 Σ 的元素称为 Σ的字符，并将签名 Σ 中的函数和关系称为Σ的基本操作。

我们还将 $\Sigma^\star$ 中的每个字符串与一个有限 Σ-结构对应起来，具体如下。

定义2. 给定一个字符串 $s \in \Sigma^\star$，令 Σ-结构（也称为 s）的论域为 s 的索引集合（我们用自然数的一个初始段表示该集合）；对于 s的索引 x，令

- $s \models \sigma(x)$ 当且仅当 s 在索引 x 处的字符是 σ，
- $s \models \mathsf{min}(x)$ 当且仅当 x 是最小索引 0，
- $s \models$ 最大值(x) 当且仅当 x 是最大的索引 $|s|-1$，
- $s \models \mathsf{S}(x) = y$ 当且仅当 $y = x+1$ 或 $x = y = |s|-1$，并且
- $s \models \mathsf{P}(x) = y$ 当且仅当 $y = x-1$ 或 $x = y = 0$。

注意，后继和前驱分别固定最大的索引和最小的索引。还需注意，有限的 Σ-结构是字符串，当且仅当对于每个 $x < |s|$，$s \models \sigma(x)$ 都对应一个唯一的字符 $\sigma \in \Sigma$。为避免歧义，我们常常将 $\Sigma^\star$ 中的字符串与其在签名 Σ 上的一阶结构混为一谈。此外，我们采用集合论惯例，将自然数与其前驱集合等同，即 $n = \{0, 1, \ldots, n-1\}$。这样，我们就可以将字符串 s 的论域与其长度 $|s|$ 等同起来。

3. Boolean monadic recursive schemes

现在我们定义布尔单子递归方案（BMRS）。这是一种最初在 [BCJO20] 中引入的编程语言。我们以纯函数式风格呈现这些方案，并为其配备标准的基于环境的大步语义。[8]

程序相对于给定的有限字符串执行。程序变量的取值范围包括两种数据类型：布尔值 bool 和字符串索引 ind。布尔单子递归方案中的每个递归函数符号都必须具有类型 ind $\to$ bool；也就是说，它们是布尔值的且为单子的（即只有一个输入变量），因此得名。

这两项要求对布尔单子递归方案的表达能力施加了相当严格的限制。例如，由于我们无法编写类型为 ind $\to$ind 的函数，这意味着我们无法对字符串索引进行算术运算，例如定位字符串的中点。由于我们无法编写类型为 ind $\times$ind $\to$ bool 的函数，

[8]我们的表述尤其受到 McCarthy [McC59]（经由 Moschovakis [Mos19]）的影响。

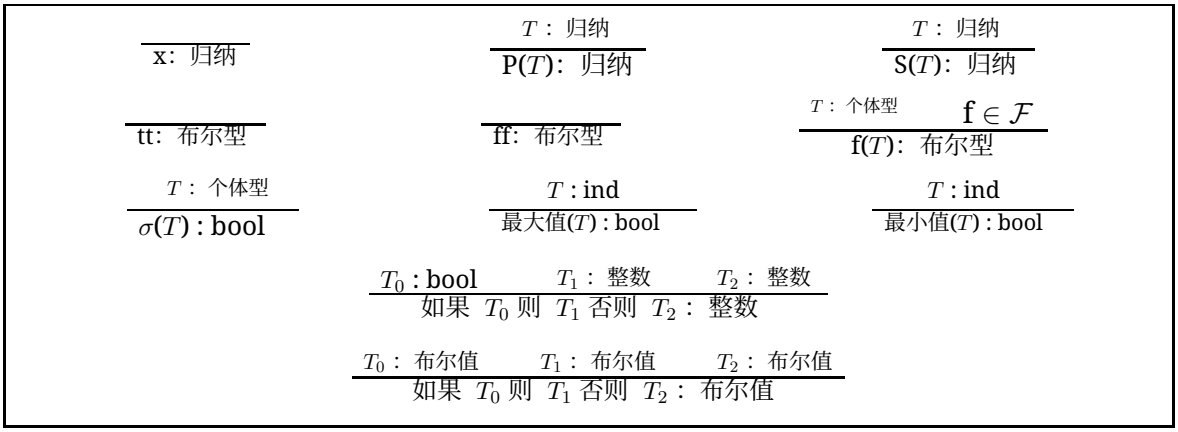


图1: Σ-项。其中, σ 遍历 Σ 的字符。

这意味着我们不能通过计算函数 ind →ind 的图关系来取巧。输入和输出类型不同这一事实，排除了嵌套递归调用。

我们首先定义项，然后定义程序，最后定义它们的语义。我们使用等宽字体表示程序语法（即程序变量和关键字——编程语言中纯粹的逻辑元素）。我们固定一个可数无限的递归函数名集合 $\mathcal{F}$，以及一个类型为 ind 的单个变量 x。

定义 3。给定一个签名 Σ, Σ-项是指任何能够由图1中的推理规则推导出的项。

定义 4。无头的 Σ-布尔单子递归方案是由有限行组成的集合

$$\begin{aligned}\mathtt{f}_0(\mathtt{x}) &= T_0\\ \mathtt{f}_1(\mathtt{x}) &= T_1\\ &\vdots\\ \mathtt{f}_k(\mathtt{x}) &= T_k\end{aligned}$$

使得对于每个 $0 \le i \le k$, T_i 是一个 Σ-项，其中不包含除（$\mathtt{f}_0$、.....、$\mathtt{f}_k$）之外的任何其他递归函数名。[9]

对于一个无头 BMRS $p = \{\mathtt{f}_i(\mathtt{x}) = T_i\}_{0\le i\le k}$, p-项指除 $(\mathtt{f}_0, \ldots, \mathtt{f}_k)$ 之外不含其他递归函数名的项。特别地，每个 T_i 都是一个 p-项。

定义5（语义）。设 Σ 为一个签名, s 为一个 Σ-字符串, x 为 s的一个索引, p 为一个无头的 Σ-BMRS, T 为一个 p-项, v 为一个类型与 T相符的值。我们根据图2中的推理规则定义五元关系 s, $x \vdash_p T \to v$ 。

最后，我们给出布尔单子递归方案的头。与 [Mos19], 的递归程序不同，我们允许存在多头程序。

定义6。对于一个签名 Σ, Σ-布尔单子递归方案由一个无头的 Σ-布尔单子递归方案 p以及一个有限的非空 p-项集合组成，这些项称为头。

[9]请注意 这是一个 set ，而不是一组 $list$ 行；顺序并不重要。然而，极其繁重的编程 i 直觉促使我们将其呈现为列表。

$$\frac{}{x \vdash \mathsf{x} \to x} \qquad \frac{}{x \vdash \mathsf{tt} \to \top} \qquad \frac{}{x \vdash \mathsf{ff} \to \bot}$$

$$\frac{x \vdash T \to v}{x \vdash \mathsf{P}(T) \to v-1} \text{（如果 } v>0\text{）} \qquad \frac{x \vdash T \to v}{x \vdash \mathsf{P}(T) \to v} \text{（如果 } v=0\text{）}$$

$$\frac{x \vdash T \to v}{x \vdash \mathsf{S}(T) \to v+1} (\text{if } v<|s|-1) \qquad \frac{x \vdash T \to v}{x \vdash \mathsf{S}(T) \to v} \text{（如果 } v=|s|-1\text{）}$$

$$\frac{x \vdash T \to v}{x \vdash \text{最小值}(T) \to \bot} \text{（如果 } v>0\text{）} \qquad \frac{x \vdash T \to v}{x \vdash \text{最大值}(T) \to \top} \text{（如果 } v=0\text{）}$$

$$\frac{x \vdash T \to v}{x \vdash \text{最大值}(T) \to \bot} \text{（如果 } v<|s|-1\text{）} \qquad \frac{x \vdash T \to v}{x \vdash \text{最大值}(T) \to \top} \text{（如果 } v=|s|-1\text{）}$$

$$\frac{x \vdash T \to v}{x \vdash \sigma\ (T) \to \top} \text{（如果 } s_v=\sigma\text{）} \qquad \frac{x \vdash T \to v}{x \vdash \sigma\ (T) \to \bot} \text{（如果 } s_v\neq\sigma\text{）}$$

$$\frac{x \vdash T \to v \qquad s, v \vdash T^{\mathtt{f}} \to w}{x \vdash \mathsf{f}\ (T) \to w}$$

$$\frac{x \vdash T_0 \to \top \qquad x \vdash T_1 \to v}{x \vdash \text{如果 } T_0\text{，则 } T_1\text{，否则 } T_2 \to v}$$

$$\frac{x \vdash T_0 \to \bot \qquad x \vdash T_2 \to v}{x \vdash \text{若 } T_0 \text{ 则 } T_1 \text{ 否则 } T_2 \to v}$$

图2：程序语义，为便于阅读省略了 s 和 p。对于递归函数名称p， $T^{\mathtt{f}}$ 是其在 p中的递归定义；对于索引 v， s_v 是 s在 v处的字符。

按照 [Mos19], , 给定一个 Σ-BMRS，我们称其底层的无头部部分为其主体。我们扩展 $s, x \vdash_p T \to v$ 的用法，使其也包括 p 这些带有头部的对象；在这种情况下，请记住，$\vdash$ 的含义并不取决于这些头部。

最后，我们定义程序的两个重要子类。

定义7。BMRS ^{P}，即前驱 BMRS，是指不包含后继函数 S 的 BMRS。类似地，BMRS ^{S}，即后继 BMRS，是指不包含前驱函数 P 的 BMRS。

4. Interpretations

定义8。设 Σ 和 Γ 为字母表。一个（m重）解释 $\pi\colon \Sigma \times m \to \Gamma$ 是一个多头 Γ-BMRS π，其中每个字符 $\sigma \in \Sigma$ 和 $i < m$都对应一个头部 $\pi(\sigma, i)$。

定义9. 若解释 $\pi\colon \Sigma \times m \to \Gamma$ 满足良定义条件，则对于每个字符串 $s \in \Gamma^\star$， $i < m$ 和 $x < |s|$， $s, x \vdash \pi(\sigma,\ i) \to \top$ 至多对应一个 $\sigma \in \Sigma$，且 $s,\ x \vdash \pi(\tau,\ i) \to \bot$ 对于每个其他 $\tau \in \Sigma$ 均成立。

若 $s,\ x \vdash \pi(\sigma,\ i) \to \top$ 恰好对应一个 σ，则它还满足严格性。

定义10. 若 $\pi\colon \Sigma \times m \to \Gamma$ 是良定义解释，则由 π 诱导的转导是函数 $[\![\pi]\!]\colon \Gamma^\star \to \Sigma^\star$，满足对每个 $s \in \Gamma^\star$,

- $|[\![\pi]\!](s)| = |J|$，其中 $J = \{(q,\ r) \in |s| \times m : (\exists \sigma \in \Sigma)\ s,\ q \models \pi(\sigma,\ r) \to \top\}$，且
- 对于每个 $x < |J|$, $[\![\pi]\!](s) \models \sigma(x) \iff s, q \models \pi(\sigma, r) \to \top$，其中 (q, r) 是 J 中具有 x 前驱的唯一元素，其中

• $|s| \times m$ 按字典序排列，其中 $|s|$ 是较重要的坐标，m 是较不重要的坐标。

在 π 为严格的这一重要特例中，这可以特化为：

• $|[\![\pi]\!](s)| = m|s|$，对每个 $s \in \Gamma^\star$ 均成立，并且

• 对每个 $x < m|s|$，$[\![\pi]\!](s) \models \sigma(x) \iff s,\ q \vdash \pi(\sigma, r) \to \top$，其中 r 和 q 分别是 $x \div m$ 的余数和商。

例如，设 $\Gamma = \{a\}$、$\Sigma = 7$、8、9 和 $m = 3$。解释 π，其中 $\pi(7, 0) = \pi(8, 1) = \pi(9, 2) = a$ (x)，定义了一个函数：它接收长度为 $|s|$ a 个字符的字符串 s，并返回一个长度为 $|s| \times 3$ 的字符串，形式为 $(789)^{|s|}$。要了解这是如何实现的，下表展示了字符串 $aaaa$ 中的每个索引在各个 m 坐标 i 处如何解释：在 m 坐标 0 处为 7，在 m 坐标 1 处为 8，在 m 坐标 2 处为 9。

	0	1	2	3
i	a	a	a	a
0	7	7	7	7
1	8	8	8	8
2	9	9	9	9

按照索引顺序先排列 $aaaa$ 的索引副本，再按照 $i < 3$ 的顺序排列，我们得到字符串 789789789789。注意，π 是严格的。

作者此前已在 [BCJO20] 中证明，保序 BMRS^{P} 解释和保序 BRMS^{S} 解释恰好描述左次序列函数和右次序列函数，即：

定理 1. 对于任意定义良好的保序 BMRS^{P}（分别为 BMRS^{S}）解释 π，$[\![\pi]\!]$ 是左次序列（分别为右次序列）函数。类似地，对于任意左次序列（分别为右次序列）函数 f，存在某个保序 BMRS^{P}（分别为 BMRS^{S}）解释 π，使得 $f = [\![\pi]\!]$。

本文解决了 [BCJO20] 中遗留的一个问题；也就是说，刻画同时存在 P 和 S 时的保序 BMRS 解释。事实证明，句法复合问题的解答几乎不费额外功夫，就能让我们得到这一结果。

继续之前，我们注意到，将一个解释代换进程序中是一种自然的操作。

定义 11. 设 p 是一个无头 Σ-BMRS，$\pi\colon \Sigma \times m \to \Gamma$ 是一个解释。对于每个 $i < m$，令 p_i^π 为从 p 得到的 Γ-BMRS，其中将 σ 的任意出现替换为 $\pi(\sigma, i)$。

如上所述，这个定义有些不精确。由于 BMRS 语言中没有 let 或 where 结构，无法直接将一个程序嵌入另一个程序，我们的意思是：取 p，在 π 的主体下方粘贴一份副本，并将 σ 的每次出现替换为与 $\pi(\sigma, i)$ 对应的头部。我们相信这样已经足够清楚。[10] 请注意，每个 p-项都是一个 p_i^π-项。

替换具有以下性质。

[10] 一个技术性说明：取两个无头程序的并集时，我们可能会重命名递归函数名称，因此不会发生意外重叠。

引理 1. 对于任意定义良好的严格解释 π：$\Sigma \times m \to \Gamma$、任意字符串 $s \in \Gamma^\star$、任意无头部 Σ-BMRS p、任意 p-项 T、任意 $x < m|s|$，以及任意布尔值 $b \in \{\top, \bot\}$，

$$[\![\pi]\!](s), x \vdash_p T \to b \iff s, q \vdash_{p^\pi_r} T \to b,$$

其中，$q < |s|$ 和 $r < m$ 分别是 $x \div m$的商和余数 .

5. Syntactic composition

现在我们转向以下问题：给定两个“可组合”的解释 π：$\Sigma \times m \to \Gamma$ 和 ρ：$\Delta \times n \to \Sigma$，我们如何定义一个解释 ω：$\Delta \times mn \to \Gamma$，使得$[\![\rho]\!] \circ [\![\pi]\!] = [\![\omega]\!]$? 这是程序语言理论中的一个一般性问题，即：如何将给定的函数语义运算拉回到程序文本本身所对应的句法运算？[11]

在大多数程序语言中，证明程序可以组合这一事实本身并不困难；其中的微妙之处（如果确实存在的话）通常在于如何使组合尽可能高效。然而，对于保序 BMRS 解释，情况并非如此。例如，假设我们想要组合上面的解释 ρ 和 π。粗略地说，ρ 使用 $\sigma(x)$ 作为基本操作，为 $\delta \in \Delta$ 定义每个函数 $\delta(x)$（“索引 x 携带字符 δ”）。类似地，π 使用 $\gamma(x)$ 作为基本操作，为 $\sigma \in \Sigma$ 定义每个函数 $\sigma(x)$，其中 $\gamma \in \Gamma$。

因此，人们很容易想要通过取 ρ，并将其中每次对 $\sigma(x)$ 的调用替换为 σ 在 π 中的定义，来定义 ω。这确实是正确的思路。问题在于，保序解释并非完全具有复合性：它们会对输入字符串使用后继函数和前驱函数，但并不要求在输出字符串上定义这些函数。

对于严格解释而言，这并不是什么大问题，因为索引后继和前驱很容易从输入字符串沿用到输出字符串。（例如，在严格的 1 重解释中，我们可以直接将输入字符串和输出字符串中的索引对应起来。）然而，对于非严格解释，情况就复杂得多了，因为它们可能会在输出字符串中跳过某些索引。简单地说，为了关联输入和输出中的索引，程序似乎必须统计跳过的索引；而在使用布尔值递归函数的语言中，这是不可能做到的。克服这一障碍正是本文的核心技术贡献。

在本节中，我们将函数复合的问题分解为若干子问题。除一个之外，这些子问题的句法版本都很常规，我们将在本节中逐一处理。最后一个子问题触及问题的核心，因此我们将留待单独一节讨论。

5.1. 四个子问题。

称一个严格有界函数为字符串函数 f，如果存在一个自然数 n，满足 $|f(x)| \leq n|x|$，对 x 中属于 f 的论域的每个元素都成立。在这种情况下，我们称 f 被 n 严格有界。当然，如果 π 是一个 n 重解释，那么 $[\![\pi]\!]$ 被 n 严格有界。类比解释，称严格有界函数 f 是严格的，如果存在一个 n，使得对f的论域中每个 x，都有 $|f(x)| = n|x|$。

我们要考虑的问题是：给定两个严格有界函数 $f : \Gamma^\star \to \Sigma^\star$ 和 $g : \Sigma^\star \to \Delta^\star$，它们分别由 n 和 m 界定，如何将它们复合起来，从而得到

[11]参见 [Jon91]，这类运算在其中被称为 *symbolic*，而非 *syntactic*。

$g \circ f \colon \Gamma^\star \to \Delta^\star$，由 nm 界定？假设我们知道如何复合严格尖锐有界函数，并希望以尽可能简单的方式将一般问题归约到严格情形。

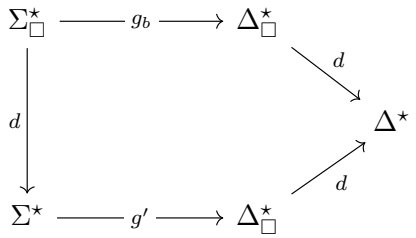
设 $\square$ 是一个既不出现在 Σ、 Δ中，也不出现在 Γ中的字符。（我们将它视为一个“空白”字符。）令 $\Sigma_\square$、 $\Delta_\square$和 $\Gamma_\square$ 分别由 Σ、 Δ和 Γ 添加这个空白字符得到。于是存在一个自然的删除映射，例如 $d_\Sigma \colon \Sigma_\square^\star \to \Sigma^\star$，它会从给定字符串中删除所有空白字符。（因此，例如， $d(ab\square a\square\square) = aba$）我们将宽松地使用记号，直接将 d_Σ、 d_Γ等写作 d 。

称 $f' \colon \Gamma^\star \to \Sigma_\square^\star$ 是 $f \colon \Gamma^\star \to \Sigma^\star$ 的严格化，当且仅当这两个函数都由同一个界严格有界、 f'是严格的，且 $d \circ f' = f$成立。任意严格有界函数都可以通过任意填充空白字符，使每个 $f(x)$的长度达到 $n|x|$，从而得到一个（不唯一的）严格化。我们或许可以通过严格有界函数 f 和 g 来得到 $g \circ f$ ，方法是

- 找到 f 和 g的两个严格化 f' 和 g' ，并且
- 将它们复合（即 $g' \circ f'$）。

这有两个问题。较轻微的问题是， g' 的陪域是 $\Delta_\square^\star$ ，而不是 $\Delta^\star$。我们只需在外侧复合 d 即可解决这个问题： $\Delta_\square^\star \to \Delta^\star$，这是对程序编码的一种简单操作。更严重的问题是， f'和 g'无法复合，因为 f'的陪域与 g' 的论域并不相同（ $f' \colon \Gamma^\star \to \Sigma_\square^\star$ ，而 $g' \colon \Sigma^\star \to \Delta_\square^\star$）。我们起初可能会天真地尝试用 $d \circ f'$替换 f'来解决这个问题，直到意识到现在我们要将（非严格的） $d \circ f'$ 与 g复合，这又让我们回到了原点。

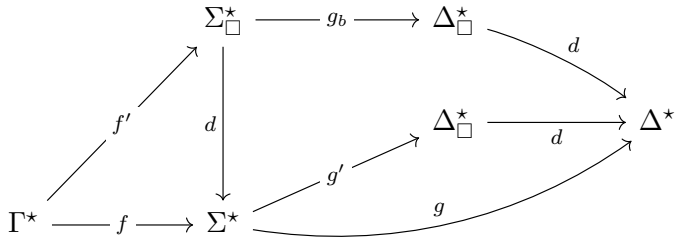
相反，我们将 $g' \colon \Sigma^\star \to \Delta_\square^\star$ 转换为另一个严格的函数 $g_b \colon \Sigma_\square^\star \to \Delta_\square^\star$ ，使得$d \circ g_b = d \circ g' \circ d$，即下图交换：



在这种情况下，

$$d \circ g_b \circ f' = d \circ g' \circ d \circ f' = g \circ f$$

这正是我们一开始想要计算的结果。图示地，



因此，我们将组合 g 与 f 的问题分解为以下子任务：
(1) 将 f 严格化为 f' ，并将 g 严格化为 g'。(2) 将 g' 提升为 g_b，我们称之为空白符扩充。(3) 复合严格函数 g_b 和 f'。

(4) 将 $(g_b \circ f')$ 去严格化为 $d \circ (g_b \circ f')$。
其中，严格化和去严格化最容易实现为对程序编码的操作，因此我们先处理它们。严格解释的复合很直接，不过要清晰地陈述并验证它，还需要一些谨慎。空白符扩充最为复杂，既需要新的想法，也需要关注技术细节，因此我们将其推迟到最后。

5.2. 严格化。这或许是对程序编码进行的最简单的变换。给定一个解释 $\pi\colon\ \Sigma \times m \to \Gamma$，我们定义 $\pi'\colon\ \Sigma_\square \times m \to \Gamma$，简单地规定：当输入字符串的某个索引处没有字符时，该索引承载空白字符 $\square$；当该索引处没有字符时，将其置于 π' 中的 π。这样做的效果是，通过在 $\square$ 中填入 $[\![\pi]\!]$ 原本为空的位置，得到 $[\![\pi']\!]$，从而有效地将其严格化。更准确地说：

定义 12。给定 $\pi\colon\ \Sigma \times m \to \Gamma$，令 $\pi'\colon\ \Sigma_\square \times m \to \Gamma$ 与 π 具有相同的主体。定义其头部如下：

对于每个 i 和 $\sigma \neq \square$，• $\pi'(\sigma, i) \equiv \pi(\sigma, i)$，并且
• $\pi'(\square, i) \equiv \bigwedge_{(\sigma \neq \square)} \neg \pi(\sigma, i)$.[12]

于是我们有：

引理 2。 若 π 定义良好，则 π' 定义良好且严格的；此外，$d \circ [\![\pi']\!] = [\![\pi]\!]$。

证明。固定 $s \in \Gamma^\star$、$i \in m$ 和 $x < |s|$。若对于某个 $\sigma \in \Sigma$，$s, x \vdash \pi(\sigma, i) \to \top$，那么仅对该 σ，$s, x \vdash \pi'(\sigma, i) \to \top$，并且 $s, x \vdash \pi'(\square, i) \to \bot$。若对于任何 $\sigma \in \Sigma$，$s, x \vdash \pi(\sigma, i) \to \top$ 都不成立，那么 π' 也同样如此；此外，$s, x \vdash \pi'(\square, i) \to \top$。这表明 π' 定义良好且严格的。

要证明 $d \circ [\![\pi']\!] = [\![\pi]\!]$，只需证明：对于每个 Γ-字符串 s 以及 $[\![\pi]\!](s)$ 的索引 x，如果 $[\![\pi]\!](s)$ 在索引 x 处带有 σ，那么 $[\![\pi']\!](s)$ 在索引 y 处带有 σ；其中 y 是 $[\![\pi']\!](s)$ 中有 x 个非空前驱的索引。

固定一个字符串 $s \in \Gamma^\star$。$[\![\pi]\!](s)$ 的索引可以与满足以下条件的有序对 (q, r) 集合 J 等同：$\in |s| \times m$，且对于某个 $\sigma \in \Sigma$，有 $s, q \models \pi(\sigma, r) \to \top$。由于 π' 是严格的，$[\![\pi]\!](s)$ 的索引可以简单地与集合 $|s| \times m$ 等同。先按 q、再按 r 对该集合进行字典序排序，并令 J 作为子集继承由此导出的顺序。

现在固定 $x < |[\![\pi]\!](s)|$。存在唯一的 $\sigma \in \Sigma$ 使得 $[\![\pi]\!](s) \models \sigma(x)$。令 (q, r) 为 J 中具有 x 个前驱的唯一元素。那么 $s, q \models \pi(\sigma, r) \to \top$，因此 $s, q \models \pi'(\sigma, r) \to \top$，这说明 $[\![\pi']\!](s) \models \sigma(mq + r)$。但 $mq + r$ 正是 $[\![\pi']\!](s)$ 中具有 x 个非空前驱的索引，这恰好就是我们要证明的。□

5.3. 去严格化。假设我们有一个定义良好、严格的解释 $\pi\colon \Sigma_\square \times m \to \Gamma$，并且希望得到一个定义良好、非严格的解释 $\pi^\dagger\colon\ \Sigma \times m \to \Gamma$，使得 $[\![\pi^\dagger]\!] = d \circ [\![\pi]\!]$。这个变换极其简单：我们只需取 π，并“忘记”每个 $\pi(\square, i)$。

定义13。给定如上所述的 π，令 $\pi^\dagger$ 具有相同的主体。对于每个 $i < m$ 和字符$\sigma \in \Sigma$，令

$$\pi^\dagger(\sigma, i) \equiv \pi(\sigma, i).$$

[12]这应被视为一系列条件语句的语法糖。

于是我们得到：

引理 3. 如果 π 是定义良好的严格解释，那么 $\pi^\dagger$ 是定义良好的；此外，$[\![\pi^\dagger]\!] = d \circ [\![\pi]\!]$。

证明。要证明 $\pi^\dagger$ 是定义良好的，我们甚至不需要 π 是严格的； $\pi^\dagger$ 从 π继承定义良好性是显而易见的。

与引理 2 的证明一样：要证明 $d \circ [\![\pi]\!] = [\![\pi^\dagger]\!]$，只需证明，对于每个 Γ-字符串 s 以及 $[\![\pi^\dagger]\!](s)$ 的索引 x，如果 $[\![\pi^\dagger]\!](s)$ 在索引 x处携带 σ，那么 $[\![\pi]\!](s)$ 在索引 y处携带 σ；其中 y 是 $[\![\pi]\!](s)$ 中具有 x 个非空前驱的索引。和之前一样，令 J 为 $|s| \times m$ 中满足以下条件的那些对 (q, r)：对于某个 $\sigma \in \Sigma$， $s, q \vdash \pi(\sigma, r) \to \top$。

固定 $s \in \Gamma^\star$ 以及 $[\![\pi^\dagger]\!](s)$ 的一个索引 x。令 σ 为 $[\![\pi^\dagger]\!](s)$ 在 x 处的字符，即 s、$q \vdash \pi(\sigma$、 $r) \to \top$，其中 $(q$、 $r)$ 是 J 中唯一具有 x 个前驱的元素。然而，$mq + r$ 是 $[\![\pi]\!](s)$ 中具有 x 个非空前驱的索引，这正是我们想要证明的结论。 □

5.4. 严格复合。假设我们有两个严格解释 $\pi\colon\ \Sigma \times m \to \Gamma$ 和 $\rho\colon\ \Delta \times n \to \Sigma$。我们如何构造一个严格解释 $\mu\colon\ \Delta \times mn \to \Gamma$，使得 $[\![\mu]\!] = [\![\rho]\!] \circ [\![\pi]\!]$?

考虑以下示例以作说明。假设 $\Gamma = \{0, 1\}$、 $\Sigma = \{a, b, c\}$、$m = 2$、 $\Delta = \{8, 9\}$ 和 $n = 3$。假设 $[\![\pi]\!](010) = abbcaa$，且 $[\![\rho]\!](abbcaa)$ =988989998998998899，即

	0	1	0
c_0	a	b	a
c_1	b	c	a

	a	b	b	c	a	a
c_0	9	9	9	9	9	8
c_1	8	8	9	9	9	9
c_2	8	9	8	8	8	9

那么我们希望得到 $[\![\mu]\!](010) = 988989998998998899$，也就是说，

	0	1	0
c_0	9	9	9
c_1	8	9	9
c_2	8	8	8
c_3	9	9	8
c_4	8	9	9
c_5	9	8	9

我们可以这样由两个旧表构造出这个新表：

	0	1	0
c_0	9	9	9
c_1	8	9	9
c_2	8	8	8
c_3	9	9	8
c_4	8	9	9
c_5	9	8	9

每个双边框方框都是一个“图块”。我们取第一个表，并将其中的 a、 b 和 c 替换为由第二个表得到的相应图块。在复合后的表中，6 个副本 c_0，.....， c_5 中，除以 3 所得的商告诉我们应查看第一个表中的哪个副本（c_0 或 c_1）

，而余数（c_0、 c_1 或 c_2）则告诉我们应查阅第二个表中的哪个副本。这引出了复合的如下定义。

定义 14。给定解释 $\pi\colon \Sigma \times m \to \Gamma$ 和 $\rho\colon \Delta \times n \to \Sigma$，定义解释 $(\rho \otimes \pi)\colon \Delta \times mn \to \Gamma$ 如下：

- $\rho \otimes \pi$ 的主体是对每个 $q \in m$， ρ^π_q的主体的并集。[13]
- 对于每个 $i \in mn$，令 q和 r 分别为 $i \div n$的整数商和余数。（于是 $r \in n$和 $q \in m$。）对于每个 $(\delta, i) \in \Delta \times mn$，令头部 $(\rho \otimes \delta)(\delta, i)$ 为 $\rho(\delta, r)^\pi_q$，即由 $\rho(\delta, r)$ 中的每个 σ 替换为 $\pi(\sigma, q)$ 所得到的项。

首先注意到，这个变换保持严格性：如果 π 和 ρ定义良好且严格，那么 $\rho \otimes \pi$也如此。接下来，我们证明正确性：

引理 4. $[\![\rho \otimes \pi]\!] = [\![\rho]\!] \circ [\![\pi]\!]$。

证明。我们知道，字符串 $[\![\rho \otimes \pi]\!](s)$ 和 $[\![\rho]\!]([\![\pi]\!](s))$ 的长度均为 $mn|s|$。对于每个 $x < mn|s|$ 和 $\delta \in \Delta$，我们必须证明

$$[\![\rho \otimes \pi]\!](s) \models \delta(x) \iff [\![\rho]\!]([\![\pi]\!](s)) \models \delta(x).$$

设 $q < |s|$ 和 $r < mn$ 分别是 $x \div mn$ 的商和余数。那么$x = mn \cdot q + r$。设 $q' < m$ 和 $r' < n$ 分别是 r除以 n所得的商和余数。那么 $r = n \cdot q' + r'$，且 $x = n \cdot (m \cdot q + q') + r'$。因此， r' 也是 $x \div n$ 的余数，而 $x \div n$ 的商 $q^\dagger$ 为 $m \cdot q + q'$。注意， $q^\dagger \div m$ 的商和余数分别为 q 和 q'。

根据 $[\![\rho \otimes \pi]\!]$的定义，

$$[\![\rho \otimes \pi]\!](s) \models \delta(x) \iff s, q \vdash (\rho \otimes \pi)(\delta, r) \to \top.$$

根据 $\otimes$的定义，$(\rho \otimes \pi)(\delta, r)$ 是 $\rho(\delta, r')^\pi_{q'}$，所以

$$[\![\rho \otimes \pi]\!](s) \models \delta(x) \iff s, q \vdash \rho(\delta, r')^\pi_{q'} \to \top.$$

另一方面，

$$[\![\rho]\!]([\![\pi]\!](s)) \models \delta(x) \iff [\![\pi]\!](s), q^\dagger \vdash \rho(\delta, r') \to \top,$$

根据 $[\![\rho]\!]$的定义。由引理 1，

$$[\![\pi]\!](s), q^\dagger \vdash \rho(\delta, r') \to \top \iff s, q \vdash \rho(\delta, r')^\pi_{q'} \to \top.$$

因此，通过复合这些等价关系，

$$[\![\rho \otimes \pi]\!](s) \models \delta(x) \iff [\![\rho]\!]([\![\pi]\!](s)) \models \delta(x),$$

这正是我们想要证明的。 □

[13]另一个技术说明：每个 ρ^π_q 都包含一份 π 的主体副本。它们可以在并集中被识别。否则，必须重命名每个分量中的递归函数符号，使它们在并集中不会被识别。

6. Blank enrichment

现在我们来解决以下问题：给定一个严格解释 $\pi\colon \Sigma_\square \times m \to \Gamma$，找出一个严格解释 $\pi_b\colon \Sigma_\square \times m \to \Gamma_\square$，使得 $d \circ [\![\pi_b]\!] = d \circ [\![\pi]\!] \circ d$。以下术语有助于理解其运作方式。对于字符串 $s \in \Gamma^\star_\square$，称 $d(s)$ 为其底层字符串。反过来，称 d-字符串 $\Gamma^\star$ 在 d 下的原像中的任意元素 s 为 s 的填充。若两个字符串 $s, t \in \Gamma^\star_\square$ 满足 $d(s) = d(t)$，则称它们为同胞字符串；也就是说，它们具有相同的底层字符串。

现在我们要构造 π_b，使得对于任意字符串 $s \in \Gamma^\star_\square$，$[\![\pi_b]\!](s)$ 和 $[\![\pi]\!](t)$ 都是同胞字符串，其中 t 是 s 的底层字符串。当然，实现这一点有很多方法。我们采用尽可能直接的方法。例如，如果 $m = 1$，我们只需在 $[\![\pi_b]\!](s)$ 中 s 为空白的位置重现 $\square$，然后在剩余的空位中填入 $[\![\pi]\!](t)$。例如，设 $\Gamma = \{0, 1\}$, $\Sigma_\square = \{a, b, \square\}$。那么我们希望将 π_b 定义为：

- 如果 $[\![\pi]\!](00) = ab$, $[\![\pi_b]\!](00\square) = ab\square$;
- 如果 $[\![\pi]\!](00) = ab$, $[\![\pi_b]\!](0\square 0) = a\square b$;
- 如果 $[\![\pi]\!](01) = b\square$, $[\![\pi_b]\!](0\square 1) = b\square\square$,
- 如果 $[\![\pi]\!](01) = b\square$, $[\![\pi_b]\!](\square 01) = \square b\square$,
- 如果 $[\![\pi]\!](010011) = aabbab$, $[\![\pi_b]\!](01\square 00\square 11) = aa\square bb\square ab$;
- 如果 $[\![\pi]\!](010011) = aabbab$, $[\![\pi_b]\!](010\square 01\square\square 1) = aab\square ba\square\square b$.

请注意，在所有这些情形中，$[\![\pi_b]\!](s)$ 和 $[\![\pi]\!](t)$ 都是同胞字符串，其中 t是 s 的底层字符串。

当 $m = 2$ 时，我们采用相同的做法，但对输入中的每个 $\square$，在输出中复制两份 $\square$。例如，如果 $[\![\pi]\!](010) = aa\square bab$，我们希望得到 $[\![\pi_b]\!](\square 01\square 0) = \square\square aa\square b\square\square ab$。输出中索引 (0, 1) 和 (6, 7) 处的 $\square\square$ 模式，分别来自输入中索引 0 和 3 处的 $\square$。其余部分则通过在剩余索引处填入 $[\![\pi]\!](010)$ 得到。

为什么要在程序编码上实现这种变换很困难？同样，简单地假设 $m = 1$。[14] 基本思路是，要确定索引 x 在$[\![\pi_b]\!](s)$中对应哪个字符，我们必须确定 x 之前有多少个非空白前驱（也就是说，$[\![\pi_b]\!](s)$中位于 x 之前且不对应 $\square$的索引数量），然后在字符串 $[\![\pi]\!](s)$中查找索引 $x^\star$。

这听起来很简单：给定 x，我们只需计算 $x^\star$，再用 π 找到它对应的字符。但布尔单子限制不允许我们把一个索引作为子程序来计算另一个索引！更糟的是，似乎不可能统计某个位置之前的空白索引数量。

我们需要做的是修改 π，使其“忽略” $\square$ 字符。每当 π检查某个索引是否为最小值或最大值时，π_b 就检查同一索引是否为最小或最大非空白索引。每当 π 取某个索引的后继或前驱时，π_b 就向前或向后移动到下一个非空白索引。在这里，BMRS 有限的递归能力已经足够：我们只需记住当前所处的状态和前进方向，并使用尾递归跳过遇到的空白索引。这样一来，我们就不必进行任何计数。

概述就到这里，下面我们更正式地看看它是如何运作的。在本节余下部分，令 $\pi\colon \Sigma_\square \times m \to \Gamma$ 为一个固定的严格解释。

[14] 全部 所有技术难点都集中在 $m = 1$ 这一情形中，但讨论起来没那么繁琐 s.

6.1. 范式。我们可以确保对 max 和 min 的所有调用在 π 中分别具有 max(x) 和 min(x) 的形式。这种构造相当刻意：只需额外定义两个递归函数 $\mathrm{f}_{\max}$(x) $\equiv$ max(x) 和 $\mathrm{f}_{\min}$(x) $\equiv$ min(x)，并将 max(T) 或 min(T) 的其他所有出现分别替换为 $\mathrm{f}_{\max}$(T) 和 $\mathrm{f}_{\min}$(T)。完全同理，我们可以确保对任意字符 $\gamma \in \Gamma$ 的所有调用都具有 γ(x) 的形式。

接下来，我们可以确保每个递归调用都具有以下三种形式之一：f(x)、f(Sx) 和 f(Px)，其中 f 是某个递归函数名。由于 BMRS 语法的限制，每个递归调用本来就都是 f(T) 的形式，其中 T 是一个索引值项，而每个索引值项都是由若干个 S 和 P 作用于 x 所构成的字符串。通过添加更多递归函数，我们可以将其化为上述三种形式。例如，f(SSx) 可以替换为 g(Sx)，其中 g(x) $\equiv$ f(Sx)。

因此，我们所做的就是证明：可以假设 π 中对布尔值原语（最大值、最小值或某个 γ）的每次调用都具有 max(x)、min(x) 和 γ(x) 的形式，而 π 中对递归函数 f 的每次调用都具有 f(x)、f(Sx) 或 f(Px) 的形式。

6.2. 程序变换。现在我们定义一种程序变换 $p \mapsto p^\star$，将无头 Γ-BMRS 变为无头 $\Gamma_\square$-BMRS；其大致思路是：对于任意字符串 $t \in \Gamma^\star$，$p^\star$ 在 t 的任意填充上的作用，都等同于 p 在 t上的作用。

在下文的定义中，对于给定字符串的“空白”或“非空白”索引，我们分别指不带或带有字符 $\square$ 的索引。

定义 15。对于任意字符串 $s \in \Gamma_\square^\star$，令 $x \mapsto x^\star$ 为从 s的非空白索引到 $d(s)$的索引的映射，其定义为 $x^\star = x - \delta$，其中 δ是 x 在s中的空白前驱数。

用两个偏映射 $x \mapsto x^S$ 和 $x \mapsto x^P$ 将 $x \mapsto x^\star$ 扩展为从 s 的所有索引到 $d(s)$的索引的映射。具体而言，$x^S = y^\star$，其中 y 是 $\geq x$中最小的非空白索引；而 $x^P = z^\star$，其中 z是 $\leq x$中最大的非空白索引。[15]

例如，若 $\Gamma = \{a, b\}$ 且 $s = a\square\square b\square a$，则 $d(s) = aba$ 且 $0^\star = 0$、$3^\star = 1$和 $5^\star = 2$。此外，$0^P = 1^P = 2^P = 0$、$3^P = 1$和 $4^P = 5^P = 2$。最后，$0^S = 0$、$1^S = 2^S = 3^S = 1$和 $4^S = 5^S = 2$。注意，所有这些映射都是单调不减的，并且 $x \mapsto x^\star$ 是从 s 的非空白索引到 $d(s)$的索引的双射。

现在考虑由下式定义的函数 maxc：

$$\mathrm{maxc}(\mathtt{x}) \equiv \mathtt{if}\ \square(\mathtt{x})\ \mathtt{then}\ \mathtt{f}(\mathtt{x})\ \mathtt{else}\ \bot$$
$$\mathtt{f}(\mathtt{x}) \equiv \mathtt{if}\ \max(\mathtt{x})\ \mathtt{then}\ \top\ \mathtt{else\ if}\ \square(\mathrm{Sx})\ \mathtt{then}\ \bot\ \mathtt{else}\ \mathtt{f}(\mathrm{Sx}).$$

于是，maxc 可以检测最大非空白索引。我们也可以类似地定义 minc。

以下 评注概括了迄今定义的映射的若干基本性质：

备注 6.1. 对于任意字符串 $s \in \Gamma_\square^\star$、其中任意非空白索引 x 以及任意字符 $\gamma \in \Gamma$，我们有：

(1) $s \models \gamma(x) \iff d(s) \models \gamma(x^\star)$, (2) $s \vdash \mathrm{maxc}(x)$
$\to \top \iff d(s) \models$ 最大值$(x^\star)$, (3) $s \vdash \mathrm{minc}(x)$
$\to \top \iff d(s) \models$ 最小值$(x^\star)$, (4) $s \vdash \mathrm{maxc}(x)$
$\to \bot \iff d(s) \models \neg$最大值$(x^\star)$, (5) $s \vdash \mathrm{minc}(x)$
$\to \bot \iff d(s) \models \neg$最小值$(x^\star)$,

[15]这些映射 可能是部分映射，因为如果没有后继或前驱，x^S 和 x^P 将定义不良
非空白索引 分别为 x。

(6) $x^S = x^P = x^\star$，以及 (7) $\mathrm{P}x^\star = (\mathrm{P}x)^P$，以及 $\mathrm{S}x^\star = (\mathrm{S}x)^S$。
最后，对于 s 的任意空白索引 x，$x^S = (\mathrm{S}x)^S$ 和 $x^P = (\mathrm{P}x)^P$。

定义16。对于每个递归函数名称 f，令 $\mathrm{f}^\star$、f^S 和 f^P 为三个不同的递归函数符号。我们定义一个变换 $T \mapsto T^\star$，将布尔值的 Γ-项（采用上述范式）映射为布尔值的 $\Gamma_\square$-项，如下所示。

- 如果 $T \equiv \top$ 或 $T \equiv \bot$，则 $T^\star \equiv T$。
- 如果对于任意字符 $\gamma \in \Gamma$，$T \equiv \gamma(\mathrm{x})$，则 $T^\star \equiv \gamma(\mathrm{x})$。
- 如果 $T \equiv \max(\mathrm{x})$ 或 $T \equiv \min(\mathrm{x})$，则分别为 $T^\star \equiv \mathrm{maxc}(\mathrm{x})$ 或 $T \equiv \mathrm{minc}(\mathrm{x})$。
- 如果 $T \equiv \mathrm{f}(\mathrm{x})$，则对于任意递归函数名称 f，有 $T^\star \equiv \mathrm{f}^\star(\mathrm{x})$。
- 若 $T \equiv \mathrm{f}(\mathrm{Sx})$，则对于任意递归函数名称 f，有 $T^\star \equiv \mathrm{f}^S(\mathrm{Sx})$。
- 若 $T \equiv \mathrm{f}(\mathrm{Px})$，则对于任意递归函数名称 f，有 $T^\star \equiv \mathrm{f}^P(\mathrm{Px})$。
- 若 $T \equiv$ 若 T_0 则 T_1 否则 T_2 则 $T^\star \equiv$ 若 $T_0^\star$ 则 $T_1^\star$ 否则 $T_2^\star$。

定义17。对于任意无头 Γ-BMRS $p = (\mathrm{f}_i(\mathrm{x}) = T_i)_{0 \le i \le k}$，通过 $(\mathrm{f}_i^\star(\mathrm{x}) = T_i^\star)_{0 \le i \le k}$ 加上对于每个 $0 \le i \le k$ 定义 $\Gamma_\square$-BMRS $p^\star$，

$$\mathtt{f}_i^S(\mathrm{x}) = \mathtt{if}\ \square(\mathrm{x})\ \mathtt{then}\ \mathtt{f}_i^S(\mathrm{Sx})\ \mathtt{else}\ T_i^\star,$$
$$\mathtt{f}_i^P(\mathrm{x}) = \mathtt{if}\ \square(\mathrm{x})\ \mathtt{then}\ \mathtt{f}_i^P(\mathrm{Px})\ \mathtt{else}\ T_i^\star.$$

定理 2。对于任意字符串 $s \in \Gamma_\square^\star$、无头 Γ-BMRS p、s 的非空白索引 x、布尔值 b项 p T，

$$d(s), x^\star \vdash_p T \to b \implies s, x \vdash_{p^\star} T^\star \to b.$$

此外，对于 s 的任意索引 x、任意递归函数符号 f，以及任意布尔值 b，如果x^S 存在，那么

$$d(s), x^S \vdash_p \mathtt{f}(\mathrm{x}) \to b \implies s, x \vdash_{p^\star} \mathtt{f}^S(\mathrm{x}) \to b,$$

如果 x^P 存在，那么

$$d(s), x^P \vdash_p \mathtt{f}(\mathrm{x}) \to b \implies s, x \vdash_{p^\star} \mathtt{f}^P(\mathrm{x}) \to b.$$

证明。固定 s、p、x、T 和 b。令 $t = d(s)$ 为 s 底层的 $\Gamma^\star$ 字符串。为便于阅读，我们省略推导符号下标。我们将 t、$\cdots \vdash \ldots$ 理解为 t、$\cdots \vdash_p \ldots$，并将 s、$\cdots \vdash \ldots$ 理解为 s、$\cdots \vdash_{p^\star} \ldots$。我们通过对 p 中计算长度进行同时归纳，证明全部三个命题。

首先，我们来证明 t、$x^\star \vdash T \to b \implies s$、$x \vdash T^\star \to b$.。假设 t、$x^\star \vdash T \to b$。若对于某个字符 $\gamma \in \Gamma$，有 $T \equiv \gamma(\mathrm{x})$，则 $T^\star \equiv \gamma(\mathrm{x})$，结论由备注 6.1 得出。若$T \equiv \max(\mathrm{x})$ 或 $\min(\mathrm{x})$，则 $T^\star \equiv \mathrm{maxc}(\mathrm{x})$ 或 $\mathrm{minc}(\mathrm{x})$，结论同样由备注 6.1 得出。

假设 $T \equiv \mathrm{f}_i(\mathrm{x})$，并令 $\mathrm{f}_i(\mathrm{x}) = T_i$ 为 p 中 f_i 的递归定义。于是$t, x^\star \vdash T_i \to b$，由归纳法，$s, x \vdash T_i^\star \to b$。但 $\mathrm{f}_i^\star(\mathrm{x}) = T_i^\star$ 是 $p^\star$ 中 $\mathrm{f}_i^\star$ 的递归定义，因此 $s, x \vdash \mathrm{f}_i^\star(\mathrm{x}) \to b$，这正是我们要证明的，因为 $T^\star \equiv \mathrm{f}_i^\star(\mathrm{x})$。

假设 $T \equiv \mathrm{f}(\mathrm{Px})$。（$T \equiv \mathrm{f}(\mathrm{Sx})$ 的情形类似。）由于 $t, x^\star \vdash \mathrm{f}(\mathrm{Px}) \to b$,$t$,$\mathrm{P}x^\star \vdash \mathrm{f}(\mathrm{x}) \to b$。进一步假设 $\mathrm{P}x$ 是 s 的一个非空白索引。根据备注 6.1$t \models \mathrm{P}x^\star = (\mathrm{P}x)^P$。因此，$t, (\mathrm{P}x)^P \vdash \mathrm{f}(\mathrm{x}) \to b$。由归纳法，$s, \mathrm{P}x \vdash \mathrm{f}^P(\mathrm{x}) \to b$，于是$s, x \vdash \mathrm{f}^P(\mathrm{Px}) \to b$，这正是我们要证明的，因为 $T^\star \equiv \mathrm{f}^P(\mathrm{Px})$。

现在假设 $y = \mathrm{P}x$ 是 s 的空白索引。那么 $t \models \mathrm{P}x^\star = y^P$（备注 6.1）。于是 t，$y^P \vdash \mathrm{f}(\mathrm{x}) \to b$；根据归纳法，$s$，$y \vdash \mathrm{f}^P(\mathrm{x}) \to b$，所以 s，$x \vdash \mathrm{f}^P(\mathrm{Px}) \to b$，这正是我们要证明的结论。

最后，假设 $T \equiv \text{if } T_0 \text{ then } T_1 \text{ else } T_2$ 且 $t,\ x^\star \vdash T_0 \to \top$。（情形 $t,\ x^\star \vdash T_0 \to \bot$ 类似，只需将 T_1 替换为 T_2。）那么 $t,\ x^\star \vdash T_1 \to b$，因此 $s,\ x \vdash T_0^\star \to \top$，且 $s,\ x \vdash T_1^\star \to b$。于是，$s,\ x \vdash T^\star \to b$。至此，第一条陈述的证明完成。

接下来，我们来证明 $t, x^S \vdash \mathsf{f(x)} \to b \implies s, x \vdash \mathsf{f}^S\mathsf{(x)} \to b$。（$t, x^P \vdash \mathsf{f(x)} \to b \implies s, x \vdash \mathsf{f}^P\mathsf{(x)} \to b$ 的证明类似。）假设 $t, x^S \vdash \mathsf{f(x)} \to b$。

首先，假设 x 是一个非空白索引，因此 $x^S = x^\star$。如果 $\mathsf{f(x)} \equiv \mathsf{f}_i\mathsf{(x)}$，那么 $t, x^\star \vdash T_i \to b$。根据归纳法，$s, x \vdash T_i^\star \to b$。因此

$$s, x \vdash \texttt{if } \square(\mathtt{x}) \texttt{ then } \mathtt{f}_i^S(\mathrm{Sx}) \texttt{ else } T_i^\star \to b,$$

这意味着 $s, x \vdash \mathsf{f}_i^S\mathsf{(x)} \to b$。

最后，假设 x 是空白字符的索引。根据备注 6.1，$t \vdash (\mathrm{S}x)^S = x^S$，因此 $t, (\mathrm{S}x)^S \vdash \mathsf{f(x)} \to b$。根据归纳法，$s, \mathrm{S}x \vdash \mathsf{f}^S\mathsf{(x)} \to b$。因此，

$$s, x \vdash \texttt{if } \square(\mathtt{x}) \texttt{ then } \mathtt{f}^S(\mathrm{Sx}) \texttt{ else } \mathtt{f}(\mathtt{x}) \to b,$$

这意味着 $s, x \vdash \mathsf{f}^S\mathsf{(x)} \to b$。证明到此结束。 □

6.3. π_b的定义。最后，我们可以根据 π定义 π_b 了。回想一下，π是类型为 $\Sigma_\square \times m \to \Gamma$的严格解释。

定义 18。如下定义解释 π_b：$\Sigma_\square \times m \to \Gamma_\square$

- π_b 的主体是 $p^\star$，其中 p 是 π的主体。
- 对于每个字符 $\sigma \in \Sigma$ 和 $i < m$，头部 $\pi_b(\sigma, i)$ 为

$$\texttt{if } \square(\mathtt{x}) \texttt{ then } \bot \texttt{ else } \pi(\sigma, i)^\star.$$

- 对于每个 $i < m$，头部 $\pi_b(\square, i)$ 为

$$\texttt{if } \square(\mathtt{x}) \texttt{ then } \top \texttt{ else } \pi(\square, i)^\star.$$

引理 5. π_b 是严格解释

证明。固定一个字符串 $s \in \Gamma_\square^\star$、$x < |s|$ 和 $i < m$。若 $s_x = \square$，则对每个 $\sigma \in \Sigma$，s、$x \vdash \pi_b(\square, i) \to \top$，且$s$、$x \vdash \pi_b(\sigma, i) \to \bot$。

否则，x 是 s的一个非空白索引。由 π的严格性可知，存在唯一的 $\sigma \in \Sigma_\square$使得 $d(s)\ x^\star \vdash \pi(\sigma, i) \to \top$，且对其他任意 $\tau \in \Sigma_\square$，$d(s)\ x^\star \vdash \pi(\tau, i) \to \bot$。由定理 2，$s$、$x \vdash \pi_b(\sigma, i) \to \top$，并且对其他任意 τ，s、$x \vdash \pi_b(\tau, i) \to \bot$。这证明了 π_b的严格性。□

最后，我们证明正确性。

定理 3. 对于任意字符串 $s \in \Gamma_\square^\star$, $d([\![\pi_b]\!](s)) = d([\![\pi]\!](d(s)))$。

证明。固定一个字符串 $s \in \Gamma_\square^\star$，并令 $t = d(s)$ 为 $\Gamma^\star$ 中的底层字符串。令 $u = [\![\pi_b]\!](s)$ 和 $v = [\![\pi]\!](t)$，使得 $|u| = m|s|$ 和 $|v| = m|t|$；我们要证明 $d(u) = d(v)$。我们不直接证明这一点，而是证明 u 可以通过填充更多的 $\square$ 得到 v，这与直接证明是等价的。

令 $X \subseteq |s|$ 为 s的非空白索引集合，使得 $|X| = |t|$ 和 $x \mapsto x^\star$ 是一个双射 $X \to |t|$。令 $Y = \{y < m|s| : y \div m \in X\}$；此处 $\div$ 表示整数除法。于是，Y可视为 u的索引集合；此外 $|Y| = |v|$。定义映射 $f : Y \to |v|$，使得对每个 $q \in X$ 和 $r < m$，$f(mq + r) = mq^\star + r$。那么，f 是 Y与 v的索引集合之间唯一的单调双射。

令 y 为 u 中不属于 Y的一个索引。令 q 和 r 分别为 $y \div m$的商和余数。那么 $q \notin X$，因此 $s \models \Box(q)$。所以 $s, q \vdash \pi_b(\Box, r) \to \top$，因此 $[\![\pi_b]\!](s) \vdash \Box(y)$。换言之，$[\![\pi_b]\!](s)$将 $\Box$ 映射到 Y之外的任意索引上。

现在设 y 是 Y 中 u 的索引，并且再次设 q 和 r 分别是除以 m 所得的商和余数。设 $z = f(y) = mq^\star + r$，并设 σ 是 v 在索引 z 处的字符，使得 $[\![\pi]\!](t) \models \sigma(z)$。那么 $t, q^\star \vdash \pi(\sigma, r) \to \top$，并且根据定理 3，$s, q \vdash \pi(\sigma, r)^\star \to \top$。根据 π_b 的定义，$s, q \vdash \pi_b(\sigma, r) \to \top$；因此，$[\![\pi_b]\!](s) \models \sigma(y)$。换言之，对于每个 $y \in Y$，$[\![\pi_b]\!](s)$ 在 y 处的字符与 $[\![\pi]\!](t)$ 在 $f(y)$ 处的字符相同。

我们所做的是将 u 的索引划分为 Y 和 $|u| \setminus Y$。在前一部分中，v 作为子字符串出现；在后一部分中，只有空白字符。因此，u 是通过填充空白字符从 v 得到的，所以 $d(u) = d(v)$。□

7. BMRS 与有理函数

现在我们可以转向论文的主要结果。首先，我们为不一定是严格解释的 BMRS 解释定义一种一般的复合运算。注意，$\otimes$同时用于严格解释和非严格解释。

定义 19. 对于两个保序 BMRS 解释 $\rho\colon \Delta \times n \to \Sigma$ 和 $\pi\colon \Sigma \times m \to \Gamma$，令

$$\rho \otimes \pi = (\rho'_b \otimes \pi')^\dagger,$$

其中 ρ'_b 取作 $(\rho')_b$。

于是我们有

定理 4. 对任意上述的 ρ 和 π，有 $[\![\rho \otimes \pi]\!] = [\![\rho]\!] \circ [\![\pi]\!]$。

证明。由引理 3，$[\![\rho \otimes \sigma]\!] = d \circ [\![\rho'_b \otimes \pi']\!]$。由引理 4，$[\![\rho'_b \otimes \pi']\!] = [\![\rho'_b]\!] \circ [\![\pi']\!]$。由定理 3，$d \circ [\![\rho'_b]\!] = d \circ [\![\rho']\!] \circ d$。因此

$$[\![\rho \otimes \sigma]\!] = d \circ [\![\rho']\!] \circ d \circ [\![\pi']\!],$$

但由引理 2，$d \circ [\![\rho']\!] = [\![\rho]\!]$ 和 $d \circ [\![\pi']\!] = [\![\pi]\!]$。因此 $[\![\rho \otimes \sigma]\!] = [\![\rho]\!] \circ [\![\sigma]\!]$。□

最后，我们得到了保序 BMRS 解释对有理函数的主要刻画。我们关键地使用了 Elgot 和 Mezei 的一个定理：每个有理函数都可以分解为一个左次序列函数与一个右次序列函数的复合。

定理 5（[EM65]）。对于每个有理函数 f，$f = g \circ h$，其中某个左次序列函数为 g，某个右次序列函数为 h。

定理 6。对于任意定义良好的保序 BMRS 解释 π，$[\![\pi]\!]$ 都是有理函数。同样，对于某个保序 BMRS 解释 π，给定有理函数 f，$f = [\![\pi]\!]$。

证明。正向结论由以下事实立即得出：BMRS 是 MSO 的一个片段，且保序的 MSO 解释定义有理函数[Boj14, Fil15]。反过来，给定任意有理函数 f，考虑其由定理 5 保证的分解 $g \circ h$。根据定理 1，存在保序 BMRS 解释 ρ 和 π，使得 $g = [\![\rho]\!]$ 和 $h = [\![\pi]\!]$。最后，由定理 4 可得 $f = [\![\rho \otimes \pi]\!]$。

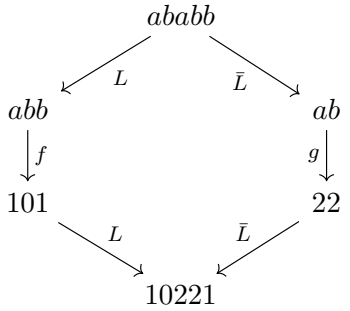
□

8. Discussion and open questions

我们将保序解释的复合分解为四个子问题：严格化、去严格化、严格复合和空白增强。前三个问题的解法非常通用，几乎可以在任何逻辑中复现。严格化只依赖于对布尔运算的封闭性，严格复合依赖于组合性（从语法上说，即能够用同类型的项替换变量），而去严格化则完全不依赖任何条件。

另一方面，句法空白符扩充似乎需要动用 BMRS 所提供的全部有限递归能力。这让我们认为，在不同逻辑上的保序解释这一背景下考察该问题十分重要。此外，我们的解决方案似乎也适用于下面这个稍微一般化的问题：

假设给定一个关系 L，它可以从任意给定字符串中选取一组索引。现在我们有两个（严格的）函数 f 和 g，希望针对 L 将它们“混合”起来，使得对于给定字符串 s，我们在 L “上”应用 f，并在L“之外”应用 g。例如，如果 $s = ababb$ 和 L 从 s中选取索引 0、1 和 4，那么将 s 限制在 L 上得到 abb，而将 s限制在其补集上得到 ab。如果 $f(abb) = 101$且 $g(ab) = 22$，那么将 f 和 g混合后得到的 L，应用于 s时结果为 10221。（见图。）



空白增强只是混合化的一种特殊情况，其中 L 是 $\square(\mathbf{x})$，f 是给定函数，而 g 是常数-$\square$ 函数。空白增强所用的技术似乎很容易推广到更一般的问题。许多其他自然的泛函都可以用混合化来表示，例如连接：给定两个函数 $f, g\colon \Sigma^\star \to \Gamma^\star$，定义函数 $\mathsf{Concat}(f, g)\colon \Sigma^\star_\square \to \Gamma^\star_\square$，使得 $u\square v \mapsto f(u)\square g(v)$，至少对恰好包含一个 $\square$ 的字符串成立。

类似连接的泛函是用于函数类（如正则函数 [AFR14]）的代数刻画中的重要示例（或组合子）。通过程序变换在句法上实现组合子（正如我们在此处对有理函数所做的那样），为我们提供了一条途径，以捕捉不同于——且可以说比——将逻辑解释编译为变换器，反之亦然，更为简洁的结果。

一个显而易见的下一步，是探究正则函数类是否存在 BMRS 刻画。由于这些函数可以由 MSO 解释（不一定保序）捕获，人们可能会猜想 BMRS 也具有同样的性质。不过，非保序的 BMRS 解释究竟是什么样的，并不十分明确：我们需要定义输出字符串上的后继函数，这意味着必须在程序中容纳非布尔值递归函数。

另一个开放方向是将 BMRS 改编用于树或图等其他数据类型，这些数据类型具有某种正则性概念。在这里，我们再次遇到一个基础性问题：如何将布尔单子范式扩展到那些难以纳入其中的数据类型。与字符串上的 BMRS 解释相比，保序 BMRS 解释的优势在于兼具简洁性与内涵表达能力。我们认为，这些特质是检验任何将 BMRS 扩展至其他论域的方案是否合格的试金石。

References

[AFR14] Rajeev Alur、Adam Freilich 和 Mukund Raghothaman。用于字符串变换的正则组合子。载于 *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)* 14, 2014，CSL-LICS '纽约，NY，美国，。计算机协会。[B̈60] J. Richard B
..
Zeitschrift für Mathematis- uchi。弱二阶算术与有限自动机。，：–。[BCJO20] Siddharth Bhaskar、JANE CHANDLEE、ADAM JARDINE 和 Christopher Oakden。布尔单子递归方案作为子序列函数的逻辑刻画。载于 Alberto Leporati、Carlos Mart
´
Language and Automata The- ın-Vide、Dana Shapira 和 Claudio Zandron（编），《计算机科学讲义》，页码 –。Springer，2020 年。[Boj14] Miko laj Boja
´
41st International Colloquium on nczyk。带起源信息的转换器。载于，*Automata Languages, and Programming (ICALP)*, 《》第 卷，页码 –。Springer，2014 年。[CJ21] JANE CHANDLEE 和 ADAM JARDINE。*Language*，97(3):485–519，2021 年 9 月。[Cou94] Bruno Courcelle。单子二阶可定义图转换：综述。*puter Science* 126 53 75, 1994 *Theoretical Com-*，：–。

[EH01] Joost Engelfriet 和 Hendrik Jan Hoogeboom。MSO可定义字符串转换与双向有限状态转换器。*ACM Transations on Computational Logic*，2:216–254，2001 年 4 月。[EM65] C. C. Elgot 和 J. E. Mezei。关于广义有限自动机定义的关系。*of Research and Development* 9 1 47 68, 1965 *IBM Journal*，( ): –。[Fil15] Emmanual Filiot。变换的逻辑—自动机联系。载于 *(ICLA)* 30 57 2015 *Logic and Its Applications*，页码 –。Springer，。

[GK04] Georg Gottlob 和 Christoph Koch。一元 Datalog 与网络信息抽取语言的表达能力。*J. ACM*，51(1):74–113，2004 年 1 月。[Jon91] Neil D. Jones。程序上的高效代数运算。载于 *University of Iowa* 393 420, ，页码 –。[McC59] 1991 John McCarthy。计算数学理论的基础。载于 P. Braffort 和 D. Hirschberg（编），《》，《》第 卷，页码 –。Elsevier，。[Mos19] Yiannis N. Moschovakis。*Abstract recursion and intrinsic complexity* 48 *Lecture Notes* ，《》第 卷。剑桥大学出版社，。[Sak09] Jacques Sakarovitch。*Elements of Automata Theory*。剑桥大学出版社，2009 年。[Sch06] Nicole Schweikardt。论单子最小不动点逻辑的表达能力。*in Logic* 2019 *Theoretical Computer Science* 350 2 325 344, 2006，( ): –。自动机、语言与编程：逻辑与语义（ICALP-B 2004）。

本作品采用知识共享署名许可协议授权。要查看该许可协议副本，请访问 https://creativecommons.org/licenses/by/4.0/，或致函知识共享，美国加利福尼亚州旧金山第二街171号300室，加利福尼亚州94105，美国；或德国柏林艾森纳赫大街2号，10777柏林，德国