字符串上的单子一阶和二阶逻辑讲义

Dino Mandrioli、Davide Martinenghi、Angelo Morzenti、Matteo Pradella 和 Matteo Rossi

电子、信息与生物工程系 (DEIB)，米兰理工大学，莱昂纳多·达·芬奇广场 32 号，20133 米兰，意大利{firstname.lastname}@polimi.it

arXiv:2301.10519v1 [cs.LO] 25 Jan 2023

1 引言

自形式语言与自动机理论创立之初，研究通过某种抽象机和通过逻辑形式体系来定义语言之间的关系，就引出了颇具挑战性的理论问题，并在系统设计与验证中催生了重要应用。此类应用的一个著名例子是经典的霍尔方法：通过一阶理论 [6] 所表达的一对前置条件和后置条件作为规约，证明类似 Pascal 的程序的正确性。

如果所涉及形式体系的计算能力达到图灵机的水平，此类验证问题便不可判定；但对于能力较弱的形式体系，例如正则语言这一重要情形，问题则可能变得可判定。最初，B¨uchi、Elgot 和 Trakhtenbrot [1,2,9]各自独立提出了一种 单子二阶逻辑，恰好定义了正则语言这一语言族，从而可以利用这一语言类的可判定性来实现自动验证。

直观而言，单子逻辑对所用谓词有一些句法限制。具体来说，只允许使用只有一个参数（即元数为 1）的谓词（序关系除外）。与通常情况一样，在一阶逻辑中只能对变量进行量化。相反，在二阶逻辑中，还可以对单子谓词（即上文所述元数为 1 的谓词）进行量化，从而得到所谓的二阶变量。

有趣的是，要用单子逻辑刻画完整的正则语言类，必须采用该逻辑的二阶版本，它比更简单的一阶版本更强大；然而，McNaughton 和 Papert [7],已证明，将逻辑限制为一阶逻辑后，用户恰好只能定义正则语言中的非计数子类——也就是说，这些语言无法“计数”某个给定子词重复出现的次数。[1]

然而，由于所需算法的复杂度难以应对，这类逻辑刻画在实践中尚未被用于实现自动验证。

[1] 例如，语言 $\{(ab)^{2^n} \mid n > 0\}$ 是计数语言。反过来，非计数语言等价于正则语言的其他一些有趣子类，例如无星语言，也就是那些能够用不使用 Kleene 星运算的正则表达式定义的语言。

后来，随着模型检验的出现，该领域取得了重大突破；它利用时序逻辑[3]对语言进行刻画。时序逻辑的表达能力与一阶逻辑相同，但其简洁性较差，却能实现更高效（尽管仍为指数级）的验证算法。

这些讲义旨在介绍一阶和二阶单子逻辑的基本内容，组织结构如下。在第 2 节中，我们讨论单子一阶逻辑，并说明其表达能力严格弱于有限状态自动机，因为它只能刻画正则语言的一个真子集——非计数语言。接着，我们在第 3 节中介绍单子二阶逻辑；从语法上说，这种逻辑是单子一阶逻辑的超集，并且恰好刻画正则语言。我们还将说明如何把自动机转换为相应的公式，反之亦然。最后，在第 4 节中，我们讨论如何将语言类的逻辑刻画（例如第 2 节和第 3 节所述的刻画）用作自动验证技术的基础。

2 字符串上的单子一阶逻辑

给定输入字母表 Σ，单子一阶逻辑 (MFO)的公式由以下元素构成：

– 一阶变量用小写字母表示（为避免与字符串混淆，采用粗体），如$\boldsymbol{x}$、$\boldsymbol{y}$等；它们的取值范围为自然数 $\mathbb{N}$。– 对于 Σ中的每个符号，都有一个单子谓词$a(\cdot)$、$b(\cdot)$等；直观地说，字符串w中$\boldsymbol{x}$位置上的字符是a，当且仅当$a(\boldsymbol{x})$在w中取真。– 自然数之间的序关系 $<$。– 通常的命题联结词和一阶量词。

更准确地说，设 $\mathcal{V}$ 为一组有限的一阶变量，Σ 为一个字母表。MFO逻辑的良构公式按照以下语法定义：

$$\varphi := a(\boldsymbol{x}) \mid \boldsymbol{x} < \boldsymbol{y} \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists \boldsymbol{x}(\varphi)$$

其中$a \in \Sigma$ 且$\boldsymbol{x}$、$\boldsymbol{y} \in \mathcal{V}$。

引入通常的预定义缩写，以表示其余的命题联结词、全称量词、算术关系 $\geq$、$\leq$、$=$、$\neq$、$>$，以及一阶变量与数值常量之间的加法和减法。

我们给出命题联结词和一阶量词的以下定义：

$$\begin{aligned}
\varphi_1 \wedge \varphi_2 &\stackrel{\text{def}}{=} \neg(\neg\varphi_1 \vee \neg\varphi_2) \\
\varphi_1 \Rightarrow \varphi_2 &\stackrel{\text{def}}{=} \neg\varphi_1 \vee \varphi_2 \\
\varphi_1 \Leftrightarrow \varphi_2 &\stackrel{\text{def}}{=} (\varphi_1 \Rightarrow \varphi_2) \wedge (\varphi_2 \Rightarrow \varphi_1) \\
\forall \boldsymbol{x}(\varphi) &\stackrel{\text{def}}{=} \neg\exists \boldsymbol{x}(\neg\varphi)
\end{aligned}$$

关系的以下定义：

$$\boldsymbol{x} \geq \boldsymbol{y} \stackrel{\text{def}}{=} \neg(\boldsymbol{x} < \boldsymbol{y})$$
$$\boldsymbol{x} \leq \boldsymbol{y} \stackrel{\text{def}}{=} \boldsymbol{y} \geq \boldsymbol{x}$$
$$\boldsymbol{x} = \boldsymbol{y} \stackrel{\text{def}}{=} \boldsymbol{x} \leq \boldsymbol{y} \wedge \boldsymbol{y} \leq \boldsymbol{x}$$
$$\boldsymbol{x} \neq \boldsymbol{y} \stackrel{\text{def}}{=} \neg(\boldsymbol{x} = \boldsymbol{y})$$
$$\boldsymbol{x} > \boldsymbol{y} \stackrel{\text{def}}{=} \boldsymbol{y} < \boldsymbol{x}$$

以及以下关于常数、自然数的后继，以及常数值的加法和减法的定义：

$\mathrm{x} = 0 \stackrel{\text{def}}{=} \forall \mathrm{y} \neg(\mathrm{y} < \mathrm{x})$ 后继$(\mathrm{x}, \mathrm{y}) \stackrel{\text{def}}{=} \mathrm{x} < \mathrm{y} \wedge \neg\exists \mathrm{z}(\mathrm{x} < \mathrm{z} \wedge \mathrm{z} < \mathrm{y})$ $\mathrm{y} = \mathrm{x} + \mathrm{k} \stackrel{\text{def}}{=} \exists \mathrm{z}_0, \ldots, \mathrm{z}_k(\mathrm{z}_0 = \mathrm{x} \wedge$ 后继$(\mathrm{z}_0, \mathrm{z}_1) \wedge$ 后继$(\mathrm{z}_1, \mathrm{z}_2) \wedge \ldots \wedge$ 后继$(\mathrm{z}_{k-1}, \mathrm{z}_k) \wedge \mathrm{y} = \mathrm{z}_k)$ $\mathrm{y} = \mathrm{x} - \mathrm{k} \stackrel{\text{def}}{=} \mathrm{x} = \mathrm{y} + \mathrm{k}$

其中k 是 $\mathbb{N}$ 中的一个常数。其他一些有用的缩写如下：

– first($\boldsymbol{x}$) 和 last($\boldsymbol{x}$) 分别表示字符串中的第一个位置和最后一个位置：
$\mathrm{first}(\boldsymbol{x}) \stackrel{\text{def}}{=} \neg\exists \boldsymbol{y}(\boldsymbol{y} < \boldsymbol{x})$，显然等价于 $\boldsymbol{x} = 0$；$\mathrm{last}(\boldsymbol{x}) \stackrel{\text{def}}{=} \neg\exists \boldsymbol{y}(\boldsymbol{y} > \boldsymbol{x})$

单子一阶逻辑公式是在字符串$w \in \Sigma^*$上解释的，并相对于赋值ν：$\mathcal{V} \to U$，其中 $U = \{0, \ldots, |w| - 1\}$，它将 $\mathcal{V}$ 映射到字符串 w 中的一个位置。请注意，如果 $w = \varepsilon$，那么对于每个变量 $\boldsymbol{x} \in \mathcal{V}$，函数 $\nu(\boldsymbol{x})$都未定义。单子一阶逻辑公式的满足关系（通常记作 $\models$）定义如下：

– $w, \nu \models \mathrm{a}(\mathrm{x})$ 当且仅当 $\nu(\mathrm{x})$ 有定义且 $w[\nu(\mathrm{x})] = \mathrm{a}$
– $w, \nu \models \mathrm{x} < \mathrm{y}$ 当且仅当 $\nu(\mathrm{x}) < \nu(\mathrm{y})$ 成立
– $w, \nu \models \neg\varphi$ 当且仅当 $w, \nu \not\models \varphi$ 成立
– $w, \nu \models \varphi_1 \vee \varphi_2$ 当且仅当 $w, \nu \models \varphi_1$ 和 $w, \nu \models \varphi_2$ 中至少一个成立
– $w, \nu \models \exists \mathrm{x}(\varphi)$ 当且仅当存在某个 $\mathrm{i} \in \{0, \ldots, |w|-1\}$，使得 $|w| > 0$ 和 $w, \nu[\mathrm{i}/\mathrm{x}] \models \varphi$ 成立

其中，$w[k]$ 表示字符串w 在位置k处的字母，而 $\nu[i/\boldsymbol{x}]$ 是将i赋给$\boldsymbol{x}$ 、其他方面与 ν 一致的映射。请注意，若$w = \varepsilon$ （因此$U = \emptyset$），则$w, \nu \not\models a(\boldsymbol{x})$对任意$a$ 和$\boldsymbol{x}$ 均成立（即并非$w, \nu \models a(\boldsymbol{x})$），并且$w, \nu \not\models \exists \boldsymbol{x}(\varphi)$对任意$\boldsymbol{x}$ 和 φ 均成立（也就是说，当被量化变量的取值范围为空集时，存在量化为假，反之全称量化为真）。

为提高可读性，只要不存在歧义风险，我们就会在记号中省略 ν ——也就是说，我们写作$w \models \varphi$，表示字符串w 满足公式 φ。

单子一阶逻辑句子是闭合的单子一阶逻辑公式。给定一个句子 φ，语言$L(\varphi)$定义为

$$L(\varphi) = \{w \in \Sigma^* \mid w \models \varphi\}.$$

我们称语言 L 可以用单子一阶逻辑表达（或用单子一阶逻辑定义，简称单子一阶逻辑可定义），当且仅当存在一个单体一阶逻辑句子 φ，使得 $L = L(\varphi)$。

2.1 示例

下面的单子一阶逻辑公式φ_{L_1}定义了语言 L_1，该语言由所有以符号 a 开头的字符串组成：

$$\varphi_{L_1} : \exists \boldsymbol{x}(\boldsymbol{x} = 0 \wedge a(\boldsymbol{x}))$$

以下公式 φ_{L_2} 定义了语言 L_2，它由所有满足以下条件的字符串构成：每个符号 a 后面都必须紧跟一个 b（注意，这些字符串不能以符号a结尾）。

$$\varphi_{L_2} : \forall \boldsymbol{x}(a(\boldsymbol{x}) \Rightarrow \exists \boldsymbol{y}(\mathrm{succ}(\boldsymbol{x}, \boldsymbol{y}) \wedge b(\boldsymbol{y})))$$

以下公式φ_{L_3}定义了语言 L_3，它由所有（非空）字符串构成，且这些字符串的最后一个符号是a。

$$\varphi_{L_3} : \exists \boldsymbol{x}(\mathrm{last}(\boldsymbol{x}) \wedge a(\boldsymbol{x}))$$

以下公式 φ_{L_4} 定义了语言 L_4，它由所有满足以下条件的字符串构成（至少包含 3 个符号）：从右往左数第三个符号是a。

$$\varphi_{L_4} : \exists \boldsymbol{x}(a(\boldsymbol{x}) \wedge \exists \boldsymbol{y}(\boldsymbol{y} = \boldsymbol{x} + 2 \wedge \mathrm{last}(\boldsymbol{y})))$$

或者，语言 L_4 也可以由以下公式 φ'_{L_4}定义：

$$\varphi'_{L_4} : \exists \boldsymbol{x}(\mathrm{last}(\boldsymbol{x}) \wedge \exists \boldsymbol{y}(\boldsymbol{y} = \boldsymbol{x} - 2 \wedge a(\boldsymbol{y})))$$

以下公式 φ_{L_ϵ}（或者，φ'_{L_ϵ}）定义了语言 L_ϵ，其中只包含空字符串（假设输入字母表至少包含符号 a）：

$$\varphi_{L_\epsilon} : \neg \exists \boldsymbol{x}(a(\boldsymbol{x}) \vee \neg a(\boldsymbol{x}))$$

$$\varphi'_{L_\epsilon} : \forall \boldsymbol{x}(a(\boldsymbol{x}) \wedge \neg a(\boldsymbol{x}))$$

公式 φ_{L_ϵ}（相应地，公式 φ'_{L_ϵ}）表示：对于语言 L_ϵ中的词 w，不存在一个使 *true* 成立的位置（相应地，在它的所有位置上*false* 都成立）。只有在位置集合为空时才会出现这种情况：根据定义，存在量化在空集上取值时为假，而全称量化在空集上取值时为真；也就是说，w 是空字符串。

最后，以下公式 $\varphi_{L_\emptyset}$ 定义了空语言（假设输入字母表 Σ 至少包含符号 a）：

$$\varphi_{L_\emptyset} : \exists \boldsymbol{x}(a(\boldsymbol{x}) \wedge \neg a(\boldsymbol{x}))$$

公式 $\varphi_{L_\emptyset}$ 自相矛盾，因为它表示存在一个位置，符号 a 既出现又不出现。没有任何字符串 w（甚至空字符串也不例外）满足 $w \models \varphi_{L_\emptyset}$，因此该公式定义了空语言。

每个单元素语言——也就是说，每个只包含一个有限长度字符串的语言——都可以平凡地用单子一阶逻辑表示。例如，考虑语言 $L_{abc} = \{abc\}$，它只包含字符串 abc。以下单子一阶逻辑公式可以轻松地定义它：

$$\varphi_{L_{abc}} : \exists \boldsymbol{x} \exists \boldsymbol{y} \exists \boldsymbol{z}(\boldsymbol{x} = 0 \wedge \boldsymbol{y} = succ(\boldsymbol{x}) \wedge \boldsymbol{z} = succ(\boldsymbol{y}) \wedge last(\boldsymbol{z}) \wedge a(\boldsymbol{x}) \wedge b(\boldsymbol{y}) \wedge c(\boldsymbol{z}))$$

2.2 单子一阶逻辑的表达能力

以下结论显然成立。

命题1。 设 L、L_1，和 L_2 是分别由单子一阶逻辑公式 φ、 φ_1 和 φ_2 定义的任意语言：

– 语言 $L_1 \cap L_2$ 由公式 $\varphi_1 \wedge \varphi_2$ 定义——即，$L(\varphi_1 \wedge \varphi_2) = L_1 \cap L_2$。– 语言 $L_1 \cup L_2$ 由公式 $\varphi_1 \vee \varphi_2$ 定义——即，$L(\varphi_1 \vee \varphi_2) = L_1 \cup L_2$。– 语言 $\bar{L}$ （L 的补集）由公式 $\neg\varphi$ 定义 —— 即，$L(\neg\varphi) = \bar{L}$。

下一个定理可由命题1推出。

定理 2。 单子一阶逻辑可定义语言的族在并集、交集和取补运算下封闭。

为了进一步探究单子一阶逻辑的表达能力，[2] 我们考察单字母表 $\Sigma = \{a\}$ 上的单子一阶逻辑可定义语言。在这个简单的情形中，MFO谓词 $a(\boldsymbol{x})$ 在任何解释中的任意位置 x 上始终为真，因此它是冗余的；每个公式都等价于一个不含谓词 $a(\cdot)$ 的公式。例如， $\exists \boldsymbol{x}(a(\boldsymbol{x}) \wedge \boldsymbol{y} < \boldsymbol{x})$ 等价于 $\exists x(\boldsymbol{y} < \boldsymbol{x})$。

接下来我们将证明，对于单字母表上的这类简单语言，某个语言是单子一阶逻辑可定义的，当且仅当它是有限的[3] 或余有限的（余有限语言是指其补集为有限集的语言）。因此，例如，简单的正则 语言 $L_{even} = \{a^{2n} \mid n \geq 0\}$ 不是 单子一阶逻辑可定义的，这证明了单子一阶逻辑的表达能力严格弱于有限状态自动机、正则文法和正则表达式。

我们证明：一个单字母表上的语言可用单子一阶逻辑表示，当且仅当它是有限或余有限的。证明分为以下几步。首先，我们注意到，若一个语言是有限或余有限的，那么它就可用单子一阶逻辑表示；这是因为，正如第 2.1 节中的语言L_{abc} 所示，单元素语言可用单子一阶逻辑表示，并且单子一阶逻辑可表示语言的族对并集和取补封闭。接下来，我们证明，若一个单字母表上的语言可用单子一阶逻辑定义，那么它就是有限或余有限的。该结论又分三步证明：

1. 我们引入一种称为 QF-MFO 的新逻辑，即 MFO 的量词自由片段；2. 我们证明，在单字母表 $\Sigma = \{a\}$ 上的每种语言都是 QF-MFO 可定义的，当且仅当它是有限的或余有限的；3. 我们证明 MFO 和 QF-MFO 这两种逻辑的表达能力相同，因为每个 MFO 公式 φ 都有一个等价的 QF-MFO 公式。

---

[2] 本节沿用 Javier Esparza 于 2019 年 2 月 13 日所著讲义自动机理论——一种算法方法 中的讨论思路（据我们所知，该讲义未经编辑）。[3] 回顾一下，有限语言是指其基数有限的语言。

为了定义 QF-MFO 逻辑，我们先引入一些额外的缩写（其中 k 是 $\mathbb{N}$ 中的一个常数）：

$$\begin{aligned}
\boldsymbol{x} < \boldsymbol{y} + k &\stackrel{\text{def}}{=} \exists \boldsymbol{z}\,(\boldsymbol{z} = \boldsymbol{y} + k \wedge \boldsymbol{x} < \boldsymbol{z}) \\
\boldsymbol{x} > \boldsymbol{y} + k &\stackrel{\text{def}}{=} \exists \boldsymbol{z}\,(\boldsymbol{z} = \boldsymbol{y} + k \wedge \boldsymbol{z} < \boldsymbol{x}) \\
\boldsymbol{x} < k &\stackrel{\text{def}}{=} \exists \boldsymbol{z}\,(\boldsymbol{z} = 0 \wedge \boldsymbol{x} < \boldsymbol{z} + k) \\
\boldsymbol{x} > k &\stackrel{\text{def}}{=} \exists \boldsymbol{z}\,(\boldsymbol{z} = 0 \wedge \boldsymbol{x} > \boldsymbol{z} + k) \\
k < last &\stackrel{\text{def}}{=} \forall \boldsymbol{x}\,(last(\boldsymbol{x}) \Rightarrow \boldsymbol{x} > k) \\
k > last &\stackrel{\text{def}}{=} \forall \boldsymbol{x}\,(last(\boldsymbol{x}) \Rightarrow \boldsymbol{x} < k)
\end{aligned}$$

定义 3（量词自由单子一阶逻辑）。量词自由单子一阶逻辑的公式由以下语法定义：

$$\varphi := \boldsymbol{x} < k \mid \boldsymbol{x} > k \mid \boldsymbol{x} < \boldsymbol{y} + k \mid \boldsymbol{x} > \boldsymbol{y} + k \mid k < last \mid k > last \mid \varphi \wedge \varphi \mid \varphi \vee \varphi$$

其中，$\boldsymbol{x}$，$\boldsymbol{y} \in \mathcal{V}$ 以及 $k \in \mathbb{N}$。

在下文中，采用一种无伤大雅的重载方式，常量 k 既表示数值 $k \in \mathbb{N}$，也表示字符串 a^k。

命题 4。当且仅当语言 L 有限或余有限时，单字母表上的每个语言 L 都可由量词自由单子一阶逻辑定义。

证明。仅证明必要性：每个量词自由单子一阶逻辑句子定义的都是有限语言或余有限语言。

设 φ 是量词自由单子一阶逻辑的一个句子。由于量词自由单子一阶逻辑是无量词的，该句子 φ 是类型为 $k < last$ 和 $k > last$ 的公式的与或组合。于是，出现以下几种情况。

– $L(k < last) = \{k + 1, k + 2, \ldots\}$ 是一种余有限语言（请记住，数字标识词，反之亦然，因此 $\{k + 1, k + 2, \ldots\}$ 与 $\{a^{k+1}, a^{k+2}, \ldots\}$ 相同）。– $L(k > last) = \{0, 1, \ldots k\}$ 是一种有限语言。– $L(\varphi_1 \vee \varphi_2) = L(\varphi_1) \cup L(\varphi_2)$。如果 $L_1 = L(\varphi_1)$ 和 $L_2 = L(\varphi_2)$ 都是有限的，那么 $L(\varphi_1 \vee \varphi_2)$ 也是有限的；如果 L_1 和 L_2 都是余有限的，那么语言 $\overline{L(\varphi_1 \vee \varphi_2)} = \overline{L_1 \cup L_2} = \overline{L_1} \cap \overline{L_2}$ 是两个有限语言之交集的补集，因此它是余有限的；如果两个语言 L_1 和 L_2 中一个是有限的，另一个是余有限的，那么 $L(\varphi_1 \vee \varphi_2)$ 是一个有限语言与一个余有限语言之交集的补集，因此它是余有限的。– $L(\varphi_1 \wedge \varphi_2) = L(\varphi_1) \cap L(\varphi_2)$。如果 $L_1 = L(\varphi_1)$ 和 $L_2 = L(\varphi_2)$ 都是有限的，那么它们的交集是有限的；如果 L_1 和 L_2 都是余有限的，那么 $\overline{L(\varphi_1 \wedge \varphi_2)} = \overline{L_1 \cap L_2} = \overline{L_1} \cup \overline{L_2} = L_1 \cup L_2$ 是两个有限语言之并集的补集，因此它是余有限的；如果两个语言 L_1 和 L_2 中一个是有限的，另一个是余有限的，那么 $L(\varphi_1 \wedge \varphi_2)$ 是一个有限语言与一个余有限语言之并集的补集，因此它是有限的。

如果部分：每个有限语言或余有限语言都可以由一个量词自由单子一阶逻辑句子定义。

如果L是有限的，那么$L = \{k_1, ...k_n\}$并且

$$\varphi_L = \varphi_{\{k_1\}} \vee \cdots \vee \varphi_{\{k_n\}} =$$

$$= (k_1 - 1 < last \wedge last < k_1 + 1) \vee \cdots \vee (k_n - 1 < last \wedge last < k_n + 1)$$

如果L是余有限的，那么它的补集$\overline{L}$是有限的，因此可由某个量词自由单子一阶逻辑公式定义。于是，若对于每个量词自由单子一阶逻辑句子 φ，都存在一个量词自由单子一阶逻辑句子（记作 φ）定义语言L，即L的补集，则L 可由量词自由单子一阶逻辑定义。这样的句子 φ 等于$neg(\varphi)$，其中函数$neg(\cdot)$由以下子句归纳定义。

- $neg(k < last) = last < k \vee \underbrace{(k - 1 < last \wedge last < k + 1)}_{last=k}$
- $neg(k > last) = k < last \vee \underbrace{(k - 1 < last \wedge last < k + 1)}_{k=last}$
- $neg(\varphi_1 \vee \varphi_2) = neg(\varphi_1) \wedge neg(\varphi_2)$
- $neg(\varphi_1 \wedge \varphi_2) = neg(\varphi_1) \vee neg(\varphi_2)$

□

命题 5。单字母表上的每个单子一阶逻辑公式 φ 都等价于某个量词自由单子一阶逻辑公式f——即 $\varphi \equiv f$。

证明。证明通过对 φ 的结构进行归纳得到。

如果 $\varphi = \boldsymbol{x} < \boldsymbol{y}$，那么 $\varphi \equiv \boldsymbol{x} < \boldsymbol{y} + 0$。

如果 $\varphi = \neg\psi$，那么可以应用归纳假设，然后利用德摩根定律和诸如 $\neg(\boldsymbol{x} < \boldsymbol{y} + k) \equiv \boldsymbol{x} \geq \boldsymbol{y} + k$ 之类的等价关系消去否定（其中 $\boldsymbol{x} \geq \boldsymbol{y} + k$ 是 $\boldsymbol{x} > \boldsymbol{y} + k - 1$ 的自然简写）。

如果 $\varphi = \varphi_1 \vee \varphi_2$，那么直接应用归纳假设。

如果 $\varphi = \exists \boldsymbol{x}\, \psi$，那么根据归纳假设，$f$ 是某个 QF-MFO 公式f，并且可以假设f处于析取范式，即$f = D_1 \vee \cdots \vee D_n$，且$\varphi \equiv \exists \boldsymbol{x} D_1 \vee \cdots \vee \exists \boldsymbol{x} D_n$；接着，我们定义一个 QF-MFO 公式集合$f_i$，使得对于每个 $1 \leq i \leq n$，$f_i \equiv \exists \boldsymbol{x} D_i$ 都成立。注意，由于f是一个 QF-MFO 公式，每个f_i都不包含对变量$\boldsymbol{x}$的量化；此外，如果 φ 是一个句子，其中也不会出现同一个变量。

每个 f_i 按如下方式构造。公式 f_i 是一个合取，由 D_i 中所有不包含变量 x 的合取项，以及下文定义的合取项组成。考察 D_i中的每一对合取项，其中一个合取项的类型为 $\mathrm{t}_1 < \mathrm{x}$，其中 $\mathrm{t}_1 = \mathrm{h}$ 或 $\mathrm{t}_1 = \mathrm{y} + \mathrm{h}$，且约束是上界（即 h 是最大的，也就是说，它是类型为 $\mathrm{y} + \mathrm{h} < \mathrm{x}$ 的约束中出现的最大值）；另一个合取项的类型为 $\mathrm{x} < \mathrm{t}_2$,，其中 $\mathrm{t}_2 = \mathrm{h}$ 或 $\mathrm{t}_2 = \mathrm{y} + \mathrm{h}$，且约束是下界（即 h 是最小的）。对于每一对这样的合取项，我们都向 f_i 中添加一个等价于 $\mathrm{t}_1 + 1 < \mathrm{t}_2$的合取项；例如，若上述两个合取项分别为 $\mathrm{z} - 4 < \mathrm{x}$ 和 $\mathrm{x} < \mathrm{y} + 3$，则添加的合取项为 $\mathrm{z} < \mathrm{y} + 6 \equiv \mathrm{z} - 3 < \mathrm{y} + 3$。注意，如果只有类型为 $\mathrm{t}_1 < \mathrm{x}$ 的合取项，而缺少类型为 $\mathrm{x} < \mathrm{t}_2$ 的合取项，那么必须将（显然为真的）合取项 $\mathrm{x} < \mathrm{last} + 1$ 视为这一对中的另一个元素，以代替后者；类似地，如果只有类型为 $\mathrm{x} < \mathrm{t}_2$ 的合取项，而

如果缺少类型为 $t_1 < \boldsymbol{x}$ 的合取项，那么必须将（显然为真的）合取项 $-1 < \boldsymbol{x}$ 视为后者的替代项。

那么，$f_i \equiv \exists \boldsymbol{x} D_i$；注意，$f_i$ 不包含变量 $\boldsymbol{x}$ 的任何出现，也不包含对该变量的任何量化。□

例 *6.* 在单子一阶逻辑公式中

$$\exists \boldsymbol{x}(\boldsymbol{x} < \boldsymbol{y} + 3 \wedge \boldsymbol{z} < \boldsymbol{x} + 4 \wedge \boldsymbol{z} < \boldsymbol{y} + 2 \wedge \boldsymbol{y} < \boldsymbol{x} + 1)$$

我们识别出约束对 $z - 4 < x$ 和 $x < y + 3$，由此得到附加合取项 $z - 3 < y + 3 \equiv z < y + 6$；我们还识别出约束对 $y - 1 < x$ 和 $x < y + 3$，由此得到附加合取项 $y < y + 3$。因此，单子一阶逻辑公式 $\exists x(x < y + 3 \wedge z < x + 4 \wedge z < y + 2 \wedge y < x + 1)$ 等价于量词自由单子一阶逻辑公式

$$\boldsymbol{z} < \boldsymbol{y} + 6 \wedge \boldsymbol{y} < \boldsymbol{y} + 3 \wedge \boldsymbol{z} < \boldsymbol{y} + 2$$

□

例 *7.* 我们给出两个量词自由单子一阶逻辑公式的例子，它们分别等价于给定的单体一阶逻辑句子。

– 单子一阶逻辑公式 $\exists x \exists y \exists z (x < y \wedge y < z)$ 定义了语言 $\{a^k \mid k \geq 3\}$。通过反复应用归纳步骤，由内向外进行，我们得到 $f_1 \equiv \exists z (x < y \wedge y < z)$，以及关于被量化变量 z 的一对约束：$y < z$ 和 $z < last + 1$。由此推出约束 $y < last$，因此 $f_1 \equiv x < y \wedge y < last$ 成立；在下一归纳步骤中，我们有 $f_1 \equiv \exists y(x < y \wedge y < last)$，以及关于被量化变量 y 的一对约束：$x < y$ 和 $y < last$，由此推出 $x + 1 < last$，并得到 $f_1 \equiv x + 1 < last$；在最后一个归纳步骤中，我们有 $f_1 \equiv \exists x (x + 1 < last)$，以及关于被量化变量 x 的一对约束：$-1 < x$ 和 $x < last - 1$。由此得到 $0 < last - 1$ and $f_1 \equiv last > 1$，因此 $last > 1$ 是与原单子一阶逻辑公式 $\exists x \exists y \exists z (x < y \wedge y < z)$ 等价的 QF-MFO 公式。– 单子一阶逻辑公式 $\exists x (\neg \exists y (x < y) \wedge x < 4)$ 定义了语言 $\{a^k \mid k \leq 4\}$。同样由内向外进行，我们有 $f_1 \equiv \exists y (x < y)$，以及关于被量化变量 y 的一对约束：$x < y$ 和 $y < last + 1$，由此推出 $x + 1 < last + 1 \equiv x < last$，并得到 $f_1 \equiv x < last$；在下一归纳步骤中，我们应用否定并得到 $f_1 \equiv \exists x (x \geq last \wedge x < 4)$，以及关于被量化变量 x 的一对约束：$last - 1 < x$ 和 $x < 4$，由此得到 $f_1 \equiv last < 4$。因此，$last < 4$ 是与原单子一阶逻辑公式 $\exists x (\neg \exists y (x < y) \wedge x < 4)$ 等价的 QF-MFO 公式。□

下述定理可由命题 4 和命题 5 轻易推出。

定理 8. 单字母表上的语言可用单子一阶逻辑表达，当且仅当它是有限或余有限的。

每个单子一阶逻辑公式都定义一个正则语言。事实上，单子一阶逻辑是第 3 节中引入的单子二阶逻辑（MSO）的一个限制；如该节所示，对于每个 MSO 句子 φ，都存在一个有限状态自动机（FSA），恰好接受由 φ 定义的语言——因此，更何况，这对每个单子一阶逻辑公式也成立。因此，我们得到以下结果（证明将在第 3.1 节给出）。

陈述 1 对于每个单子一阶逻辑句子 φ，都存在一个有限状态自动机 $\mathcal{A}$，使得 $L(\varphi) = L(\mathcal{A})$。

然而，MFO逻辑的表达能力严格弱于有限状态自动机（简称 FSA），因为并非所有正则语言都能用 MFO 表达。事实上，根据定理 8，前面定义的正则语言 L_{even} 包含的恰好是字母表 $\Sigma = \{a\}$ 上长度为偶数的字符串，因此它既非有限语言，也非余有限语言；如下一条推论所述，它不能用 MFO 表达。

推论 9. 不存在定义语言 L 的单体一阶逻辑句子 φ （即满足 $L(\varphi) = L_{even}$ 的句子）。

由陈述 1 和推论 9 可立即得到以下结果：注意到很容易定义一个有限状态自动机 $\mathcal{A}$，使得 $L(\mathcal{A}) = L_{\text{even}}$。

定理 10. 单子一阶逻辑的表达能力严格弱于有限状态自动机。

另一方面，正如下述定理所述，能够通过单子一阶逻辑公式定义的语言集合对于所谓的“Kleene星号”运算并不封闭。

定理 11. 能够由 *MFO* 句子定义的语言集合对于该 * 运算并不封闭。

证明. 要证明这一结论，只需指出以下单子一阶逻辑公式 $\varphi_{L_{aa}}$定义了语言 $L_{aa} = \{aa\}$（即只包含字符串 aa 的语言）：

$$\varphi_{L_{aa}} : \exists \boldsymbol{x} \exists \boldsymbol{y}(\boldsymbol{x} = 0 \wedge \boldsymbol{y} = \boldsymbol{x} + 1 \wedge a(\boldsymbol{x}) \wedge a(\boldsymbol{y}) \wedge \text{last}(\boldsymbol{y}))$$

并且 $L_{\text{even}} = L_{aa}^*$。 □

根据定理 2 和定理 11，可以证明 [7]单子一阶逻辑只能表达所谓的“无星”语言——也就是说，这些语言可以通过对有限语言进行并集、交集、取补和连接得到。

例如，第 2.1 节的语言 L_3 可以通过以下方式，由有限语言$L_3' = \emptyset$和 $L_3'' = \{a\}$得到：前者不包含任何字符串（因此其基数为 0），后者只包含字符串a （因此其基数为 1）。

$$L_3 = (\neg L_3') \cdot L_3''$$

再举一个例子，语言 $L_{\exists a}$由所有至少包含一个a的字符串组成，可以按以下方式定义：

$$L_{\exists a} = (\neg L_3') \cdot L_3'' \cdot (\neg L_3')$$

以及由所有恰好包含一个 a 的字符串组成的语言 $L_{\exists! a}$，可定义如下：

$$L_{\exists! a} = (\neg L_{\exists a}) \cdot L_3'' \cdot (\neg L_{\exists a})$$

3 字符串上的序关系单子二阶逻辑

B¨uchi 等人 [8], 定义的单子二阶逻辑（MSO）公式由第 2 节中定义的 MFO 逻辑元素构成，此外还包括：

– 二阶变量用大写粗体字母表示，如$\boldsymbol{X}$、$\boldsymbol{Y}$ 等，其取值范围是自然数的集合。

更准确地说，设 Σ 为一个输入字母表，$\mathcal{V}_1$ 为一组一阶变量，$\mathcal{V}_2$ 为一组二阶（或集合）变量。单子二阶逻辑的良构公式按照以下语法定义：

$$\varphi := a(\boldsymbol{x}) \mid \boldsymbol{X}(\boldsymbol{x}) \mid \boldsymbol{x} < \boldsymbol{y} \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists \boldsymbol{x}(\varphi) \mid \exists \boldsymbol{X}(\varphi)$$

其中，$a \in \Sigma$，$\boldsymbol{x}$、$\boldsymbol{y} \in \mathcal{V}_1$，且$\boldsymbol{X} \in \mathcal{V}_2$。

当然，第 2 节中引入的所有缩写仍然有效。我们还引入以下额外缩写：

$$\begin{aligned}
\boldsymbol{x} \in \boldsymbol{X} &\stackrel{\text{def}}{=} \boldsymbol{X}(\boldsymbol{x}) \\
\boldsymbol{X} \subseteq \boldsymbol{Y} &\stackrel{\text{def}}{=} \forall \boldsymbol{x}(\boldsymbol{x} \in \boldsymbol{X} \Rightarrow \boldsymbol{x} \in \boldsymbol{Y}) \\
\boldsymbol{X} = \boldsymbol{Y} &\stackrel{\text{def}}{=} (\boldsymbol{X} \subseteq \boldsymbol{Y}) \wedge (\boldsymbol{Y} \subseteq \boldsymbol{X}) \\
\boldsymbol{X} \neq \boldsymbol{Y} &\stackrel{\text{def}}{=} \neg(\boldsymbol{X} = \boldsymbol{Y})
\end{aligned}$$

其中$\boldsymbol{x}$、$\boldsymbol{y}$、$\boldsymbol{X}$ 与前文相同，而$\boldsymbol{Y} \in \mathcal{V}_2$。

MSO公式在字符串$w \in \Sigma^*$上解释，并相对于赋值ν_1：$\mathcal{V}_1 \to \{0, \ldots\ldots |w|-1\}$和 ν_2：$\mathcal{V}_2 \to \mathcal{P}(\{0, \ldots\ldots |w|-1\})$。注意，与单子一阶逻辑公式的赋值$\nu$ 类似，ν_1 将 $\mathcal{V}_1$ 的每个一阶变量映射到字符串w中的一个位置。而赋值 ν_2则将 $\mathcal{V}_2$ 的每个二阶变量映射到字符串w中位置的集合 。

于是，MSO公式的满足关系 $\models$ 定义如下：

– $\mathrm{w}, \nu_1, \nu_2 \models \mathrm{a(x)}$ 当且仅当 ν_1(x)有定义，且$\mathrm{w}[\nu_1(\mathrm{x})] = \mathrm{a}$ – $\mathrm{w}, \nu_1, \nu_2 \models \mathrm{X(x)}$ 当且仅当 $\nu_1(\mathrm{x}) \in \nu_2(\mathrm{X})$成立 – $\mathrm{w}, \nu_1, \nu_2 \models \mathrm{x} < \mathrm{y}$ 当且仅当 $\nu_1(\mathrm{x}) < \nu_1(\mathrm{y})$成立 – $\mathrm{w}, \nu_1, \nu_2 \models \neg\varphi$ 当且仅当$\mathrm{w}, \nu_1, \nu_2 \not\models \varphi$ 成立 – $\mathrm{w}, \nu_1, \nu_2 \models \varphi_1 \vee \varphi_2$ 当且仅当$\mathrm{w}, \nu_1, \nu_2 \models \varphi_1$ 和$\mathrm{w}, \nu_1, \nu_2 \models \varphi_2$中至少有一个成立 – $\mathrm{w}, \nu_1, \nu_2 \models \exists \mathrm{x}(\varphi)$ 当且仅当 $|\mathrm{w}| > 0$，且存在某个$\mathrm{i} \in \{0, \ldots\ldots |\mathrm{w}| - 1\}$，使得$\mathrm{w}, \nu_1[\mathrm{i/x}], \nu_2 \models \varphi$成立 – $\mathrm{w}, \nu_1, \nu_2 \models \exists \mathrm{X}(\varphi)$ 当且仅当 $|\mathrm{w}| > 0$，且存在某个$\mathrm{S} \subseteq \{0 \ldots\ldots |\mathrm{w}| - 1\}$，使得$\mathrm{w}, \nu_1, \nu_2[\mathrm{S/X}] \models \varphi$成立

其中，$w[k]$ 和 $\nu[i/\boldsymbol{x}]$ 与第 2 节中的相同，而 $\nu_2[S/\boldsymbol{X}]$ 将 S 赋给 $\boldsymbol{X}$，其他方面则与 ν_2一致。

为提高可读性，只要不存在歧义风险，我们就会省略记号中的 v_1, v_2，并用 $w \models \varphi$ 表示字符串 w 满足 MSO公式 φ。

MFO逻辑中对句子 和由句子 φ 定义的语言$L(\varphi)$的定义同样适用。

例 *12.* 下面的 MSO 公式 φ_{even} 定义了L_{even}，该语言在第 2.2 节中引入。

$$\varphi_{\text{even}} : \exists \boldsymbol{P}\left(\forall \boldsymbol{x}\left(\begin{array}{l}(\boldsymbol{x}=0 \Rightarrow \neg \boldsymbol{P}(x)) \\ \wedge \\ \forall \boldsymbol{y}(\boldsymbol{y}=\boldsymbol{x}+1 \Rightarrow(\neg \boldsymbol{P}(\boldsymbol{x}) \Leftrightarrow \boldsymbol{P}(\boldsymbol{y}))) \\ \wedge \\ a(\boldsymbol{x}) \\ \wedge \\ (\text{last}(\boldsymbol{x}) \Rightarrow \boldsymbol{P}(\boldsymbol{x}))\end{array}\right)\right)$$

公式 φ_{even} 引入了一个二阶变量$\boldsymbol{P}$，它恰好标记字符串w中的所有偶数位置。更准确地说，w 的第一个位置（按惯例为 0）不是偶数；事实上，公式 φ_{even} 的第一个合取项指出，当 $\boldsymbol{x}$ 为 0 时，$\boldsymbol{P}(x)$不成立。此外， φ_{even} 中的第二个合取项指出，若$\boldsymbol{x}$ 之后的下一个位置存在（即满足$\boldsymbol{y} = \boldsymbol{x}+1$ 的位置$\boldsymbol{y}$），那么该位置为偶数（即$\boldsymbol{P}(\boldsymbol{y})$在该处成立），当且仅当$\boldsymbol{x}$ 位置为奇数；因此，由于第一个位置为奇数，第二个位置为偶数，第三个位置为奇数，第四个位置为偶数，依此类推。第三个合取项指出，对每个位置$\boldsymbol{x}$，$a(\boldsymbol{x})$都成立（即每个位置上都出现a）。最后一个合取项指出，字符串中的最后一个位置必须为偶数。□

3.1 单子二阶逻辑的表达能力

由于每个单子一阶逻辑公式也是一个 MSO 公式，而 MSO 公式φ_{even} 在示例 12 中引入，定义了语言L_{even}；根据命题9，该语言无法由单子一阶逻辑公式定义，因此我们得到以下显然的结果。

定理13。 单子二阶逻辑的表达能力严格强于 *MFO* 逻辑。

事实上，B¨uchi 等人最初的开创性成果表明，与 MFO 逻辑不同，MSO 确实具有与有限状态自动机相同的表达能力，具体如下述定理所述。

定理14。 语言 L 是正则的，当且仅当存在单子二阶逻辑中的一个句子 φ，使得 $L = L(\varphi)$。

在证明定理14之前，我们指出，由于对于每个有限状态自动机，都存在一个与之等价的 MSO 公式——反之亦然——MSO 具备有限状态自动机的所有封闭性，具体如下述推论所述。

推论15。 可由*MSO*公式定义的语言集合在并集、交集、取补和 *Kleene* 星运算下封闭。

定理14的证明是构造性的，也就是说，它提供了一种算法过程：对于给定的有限状态自动机 $\mathcal{A}$，构造一个等价的单子二阶逻辑句子 $\varphi_{\mathcal{A}}$，反之亦然。接下来，我们将直观地解释这一构造，并请读者参阅例如 [8]，以了解完整而详细的证明。

从有限状态自动机到单子二阶逻辑

这一构造的关键思想是：对于有限状态自动机 $\mathcal{A}$ 的每个状态 q，使用一个二阶变量 $\boldsymbol{X}_q$，其取值为所有这样的字符所在位置的集合：$\mathcal{A}$ 可能在从状态 q 出发的转移中读取这些字符。

不失一般性，我们假设 $\mathcal{A}$ 的状态集 Q 为 $\{0, 1, 。。, m\}$，其中 m 为某个数，0 表示初始状态。接着，我们将有限状态自动机 $\mathcal{A}$ 识别 L（即满足 $L = L(\mathcal{A})$）的定义编码为若干子句的合取，每个子句描述 $\mathcal{A}$ 定义的一部分：

– 我们引入一个公式来描述 δ $\mathcal{A}$ 的转移关系，其中每个转移 $\delta(q_i, a) = q_j$ 都对应自动机的一个析取项：$\forall x, y \left(y = x + 1 \Rightarrow \vee_{\delta(q_i,a)=q_j} \left(x \in X_i \wedge a(x) \wedge y \in X_j\right)\right)$。– 机器从状态 0 开始这一事实由公式 $\forall x(x = 0 \Rightarrow x \in X_0)$ 描述。– 由于自动机不可能同时处于两个不同的状态 i、j，因此，对于每一对不同的二阶变量 X_i 和 X_j，我们引入公式 $\neg\exists y(y \in X_i \wedge y \in X_j)$。– 自动机的接受条件——即 $\delta(q_i,a) \in F$——由公式 $\forall x \left(last(x) \Rightarrow \vee_{\delta(q_i,a)\in F} (x \in X_i \wedge a(x))\right)$ 表示。

最后，与自动机 $\mathcal{A}$对应的 MSO公式 $\varphi_{\mathcal{A}}$ 是以下句子

$$\varphi_{\mathcal{A}} : \exists \boldsymbol{X}_0, \boldsymbol{X}_1, \ldots \boldsymbol{X}_m(\varphi)$$

其中 φ 是上述所有子句的合取。

不难证明，满足公式 $\varphi_{\mathcal{A}}$ 的字符串集合恰好是L。

例 *16.* 考虑图 1 中所示的有限状态自动机 $\mathcal{A}_{ex}$。对应的单子二阶逻辑公

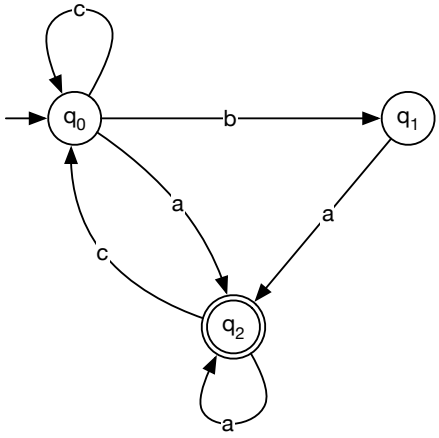


图 1：有限状态自动机 $\mathcal{A}_{ex}$。

式 $\varphi_{\mathcal{A}_{\text{ex}}}$ 按照上述规则构造如下：

$$\varphi_{\mathcal{A}_{\text{ex}}} : \exists \boldsymbol{X}_0, \boldsymbol{X}_1, \boldsymbol{X}_2 \left(\begin{array}{l} \forall \boldsymbol{x}, \boldsymbol{y} \left(\boldsymbol{y} = \boldsymbol{x} + 1 \Rightarrow \left(\begin{array}{ll} \boldsymbol{x} \in \boldsymbol{X}_0 \wedge c(\boldsymbol{x}) \wedge \boldsymbol{y} \in \boldsymbol{X}_0 & \vee \\ \boldsymbol{x} \in \boldsymbol{X}_0 \wedge b(\boldsymbol{x}) \wedge \boldsymbol{y} \in \boldsymbol{X}_1 & \vee \\ \boldsymbol{x} \in \boldsymbol{X}_0 \wedge a(\boldsymbol{x}) \wedge \boldsymbol{y} \in \boldsymbol{X}_2 & \vee \\ \boldsymbol{x} \in \boldsymbol{X}_1 \wedge a(\boldsymbol{x}) \wedge \boldsymbol{y} \in \boldsymbol{X}_2 & \vee \\ \boldsymbol{x} \in \boldsymbol{X}_2 \wedge c(\boldsymbol{x}) \wedge \boldsymbol{y} \in \boldsymbol{X}_0 & \vee \\ \boldsymbol{x} \in \boldsymbol{X}_2 \wedge a(\boldsymbol{x}) \wedge \boldsymbol{y} \in \boldsymbol{X}_2 & \end{array} \right) \right) \\ \wedge \\ \forall \boldsymbol{x} (\boldsymbol{x} = 0 \Rightarrow \boldsymbol{x} \in \boldsymbol{X}_0) \\ \wedge \\ \neg \exists \boldsymbol{y} (\boldsymbol{y} \in \boldsymbol{X}_0 \wedge \boldsymbol{y} \in \boldsymbol{X}_1) \wedge \\ \neg \exists \boldsymbol{y} (\boldsymbol{y} \in \boldsymbol{X}_0 \wedge \boldsymbol{y} \in \boldsymbol{X}_2) \wedge \\ \neg \exists \boldsymbol{y} (\boldsymbol{y} \in \boldsymbol{X}_1 \wedge \boldsymbol{y} \in \boldsymbol{X}_2) \\ \wedge \\ \forall \boldsymbol{x} \left(\text{last}(\boldsymbol{x}) \Rightarrow \left(\begin{array}{l} (\boldsymbol{X}_0(\boldsymbol{x}) \wedge a(\boldsymbol{x})) \\ \vee \\ (\boldsymbol{X}_1(\boldsymbol{x}) \wedge a(\boldsymbol{x})) \\ \vee \\ (\boldsymbol{X}_2(\boldsymbol{x}) \wedge a(\boldsymbol{x})) \end{array} \right) \right) \end{array} \right)$$

其中，第一个子句刻画自动机的转移关系；第二个子句形式化其初始状态；接下来的三个合取项陈述状态互斥；最后一个子句刻画接受条件。□

备注 *17.* 通过 MSO公式 $\varphi_{\mathcal{A}}$ 对有限状态自动机 $\mathcal{A}$ 进行形式化，尤其是刻画接受条件的子句，假定被接受的字符串至少包含一个符号（也就是说，它们不是空字符串）。要在 MSO 中形式化一个也接受空字符串的有限状态自动机 $\mathcal{A}_\epsilon$，只需在相应的公式 $\varphi_{\mathcal{A}_\epsilon}$ 中加入一个覆盖空字符串情形的析取项，具体如下：

$$\varphi_{\mathcal{A}_\epsilon} : \exists \boldsymbol{X}_0, \boldsymbol{X}_1, \ldots \boldsymbol{X}_m (\varphi) \vee \varphi_{L_\epsilon}$$

其中，φ_{L_ϵ} 是第 2.1 节中引入的、用于刻画空字符串的公式。

从单子二阶逻辑到有限状态自动机

反方向的构造在文献中已有多种版本。这里我们沿着 [8] 的思路总结其主要步骤。首先，将 MSO 句子转换为一种标准形式，该形式只使用二阶变量（不允许使用一阶变量）、 $\subseteq$ 谓词，以及变量$\boldsymbol{W}_a$；对每个$a \in \Sigma$，该变量表示词中包含字符a的所有位置构成的集合。此外，我们使用 Succ，其含义与 succ 相同，并且其参数是单元素集合上的二阶变量。这种更简单（但等价）的逻辑由以下语法定义：

$$\varphi := \boldsymbol{X} \subseteq \boldsymbol{W}_a \mid \boldsymbol{X} \subseteq \boldsymbol{Y} \mid \text{Succ}(\boldsymbol{X}, \boldsymbol{Y}) \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists \boldsymbol{X}(\varphi).$$

与之前一样，我们也使用标准缩写，例如， $\wedge$、 $\forall$、 $=$。要将一阶变量转换为二阶变量，我们需要说明一个（二阶）变量是单元素集合。因此我们引入以下缩写：

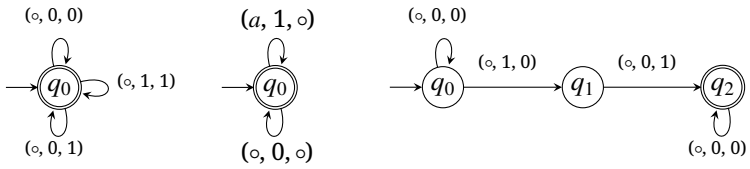
$$\mathrm{Sing}(\boldsymbol{X}) \stackrel{\text{def}}{=} \exists \boldsymbol{Y}(\boldsymbol{Y} \subseteq \boldsymbol{X} \wedge \boldsymbol{Y} \neq \boldsymbol{X} \wedge \neg \exists \boldsymbol{Z}(\boldsymbol{Z} \subseteq \boldsymbol{X} \wedge \boldsymbol{Z} \neq \boldsymbol{Y} \wedge \boldsymbol{Z} \neq \boldsymbol{X}))$$

然后，在下面的转换中，Succ(X, Y) 总是与 Sing(X) $\wedge$Sing(Y) 合取，因此当 X 或 Y 不是单元素集合时，所得公式为假。接下来的一步是归纳构造等价自动机。具体做法是为每个基本子公式关联一个自动机，并根据整体公式的结构将这些自动机组合起来。这种归纳方法需要使用开公式，即其中含有自由变量的公式。出于技术原因，对于这类公式，我们将考虑字母表为 $\Sigma \times \{0,1\}^k$ 上的词，其中 k 是自由变量的数量；在从单子二阶逻辑转换为有限状态自动机的后续步骤中，字母表将恢复为 Σ。因此，如果公式中使用的自由变量为 $\mathrm{X}_1, \mathrm{X}_2, \ldots \mathrm{X}_k$，那么，举例来说，第 j 个分量的值为 1，表示所考虑的位置属于 X_j（也就是说，二阶变量 X_j 表示一个一阶变量，其取值为所考虑的位置）；值为 0 则表示相反。例如，若 $\mathrm{w} = (\mathrm{b}, 1, 0)(\mathrm{a}, 0, 0)(\mathrm{a}, 0, 1)$，则 $\mathrm{w} \models \mathrm{X}_2 \subseteq \mathrm{W}_a$，$\mathrm{w} \models \mathrm{X}_1 \subseteq \mathrm{W}_b$，其中 X_1和 X_2 是单元素集合，分别表示字符串 w 中值为 0 和 2 的位置（即一阶变量）。

公式转换

1. 一阶变量按以下方式转换： $\exists \boldsymbol{x}(\varphi(\boldsymbol{x}))$ 变为$\exists \boldsymbol{X}(\mathrm{Sing}(\boldsymbol{X}) \wedge \varphi'(\boldsymbol{X}))$，其中 φ'是 φ 的转换结果，而 $\boldsymbol{X}$是一个全新的变量，未在其他地方出现。2. 形如 $a(\boldsymbol{x})$、$\mathrm{succ}(\boldsymbol{x}, \boldsymbol{y})$ 的子公式分别转换为 $\boldsymbol{X} \subseteq \boldsymbol{W}_a$、$\mathrm{Succ}(\boldsymbol{X}, \boldsymbol{Y})$。3. 其他部分保持不变。

自动机的归纳构造为简单起见，我们假设 $\Sigma = \{a,b\}$，并且 $k = 2$，即公式中使用两个变量。此外，在自动机的转移标签中，我们用快捷符号 $\circ$表示所有可能的取值。



(a) (b) (c)

图 2：从单子二阶逻辑到有限状态自动机的构造所用的自动机。

– 公式 $X_1 \subseteq X_2$ 被转换为一个自动机，该自动机检查：只有在 X_2分量也有 1 的位置，X_1 分量才有 1（图2 (a)）。– 公式 $X_1 \subseteq W_a$ 类似：自动机检查 X_1 分量中标记为 1 的位置是否带有符号 a（图2 (b)）。– 公式 $\mathrm{Succ}(X_1,X_2)$考虑两个单元素集合，并检查分量 X_1 中的 1 是否紧接着分量 X_2 中的 1（图2 (c)）。– 由 $\neg$ 和 $\vee$ 归纳构造的公式，分别可由正则语言关于补集和并集的闭包性质处理。– 对于形如 $\exists X(\varphi)$ 的公式，我们使用字母表投影的闭包性质；例如，我们可以从一个输入字母表为 $\Sigma \times \{0,1\}^2$、对应公式 $\varphi(X_1,X_2)$ 的自动机开始，并可能需要为公式$\exists X_1(\varphi(X_1, X_2))$ 定义一个自动机。但此时字母表为 $\Sigma \times \{0,1\}$，其中最后一个分量表示唯一剩下的自由变量，即 X_2。自动机 $\mathcal{A}_\exists$ 通过以公式 $\varphi(X_1,X_2)$ 对应的自动机为基础构造得到，并将转移标签从 (a, 0, 0) 和 (a, 1, 0) 改为 (a, 0)；将 (a, 0, 1) 和 (a, 1, 1) 改为 (a, 1)，对含 b 的标签也作同样处理。其思路是，这个自动机在读取输入时会非确定性地“猜测”量化分量（即 X_1），所得词 $w \in (\Sigma \times \{0,1\}^2)^*$ 满足 $w \models \varphi(X_1, X_2)$。因此，$\mathcal{A}_\exists$识别 $\exists X_1(\varphi(X_1,X_2))$。

关于逻辑公式与所构造自动机等价性的完整证明，请参阅现有文献。下面通过一个例子说明上述构造的原理。

例*18.* 考虑语言 $L = \{a,b\}^*aa\{a,b\}^*$：它由满足以下公式的字符串组成：

$\varphi_L = \exists \boldsymbol{x} \exists \boldsymbol{y}(\mathrm{Succ}(\boldsymbol{x},\boldsymbol{y}) \wedge a(\boldsymbol{x}) \wedge a(\boldsymbol{y}))$.

如前所述，首先我们将这个公式转换为仅使用二阶变量的形式：$\varphi'_L = \exists \boldsymbol{X}, \boldsymbol{Y}(\mathrm{Sing}(\boldsymbol{X}) \wedge \mathrm{Sing}(\boldsymbol{Y}) \wedge \mathrm{Succ}(\boldsymbol{X},\boldsymbol{Y}) \wedge \boldsymbol{X} \subseteq \boldsymbol{W}_a \wedge \boldsymbol{Y} \subseteq \boldsymbol{W}_a)$。

$\mathrm{Sing}(\boldsymbol{X})$ 和 $\mathrm{Sing}(\boldsymbol{Y})$ 的自动机如图3所示；也可以展开 Sing 的定义，然后对量化变量进行投影来得到它们。

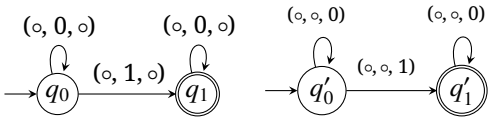


图 3：$\mathrm{Sing}(\boldsymbol{X})$ 和 $\mathrm{Sing}(\boldsymbol{Y})$ 的自动机。

通过对 $\mathrm{Sing}(\boldsymbol{X})$、$\mathrm{Sing}(\boldsymbol{Y})$ 和 $\mathrm{Succ}(\boldsymbol{X},\boldsymbol{Y})$ 对应的自动机求交，并采用常规的笛卡尔积自动机构造方法（此处未展示构造细节），我们得到一个自动机，它与我们定义的那个自动机完全相同

用于翻译公式 Succ(X_1,X_2) 的自动机，其中 X 在此处充当 X_1 的角色，而 Y 充当 X_2 的角色。将它与 $X \subseteq W_a$ 和$Y \subseteq W_a$ 对应的自动机求交，便得到图4所示的自动机。

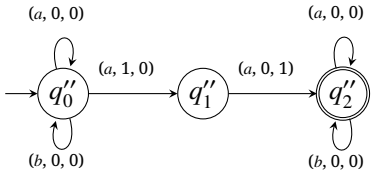


图 4：用于合取 Sing(X)、Sing(Y)、Succ(X,Y)、$X \subseteq W_a$、$Y \subseteq W_a$ 的自动机。

最后，通过对被量化变量 X 和Y 进行投影，我们得到L的自动机，如图 5 所示。□

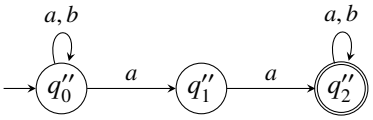


图 5：$L = \{a,b\}^* aa \{a,b\}^*$的自动机。

4 讨论

对语言类的逻辑刻画，加上相关包含问题（即检查一个语言是否是该类中另一个语言的子集）的可判定性，是实现自动验证技术的主要途径。假设逻辑形式体系 L 与自动机族 A 递归等价，那么可以用 L 的公式 $\varphi_\wp$ 来描述给定系统的需求，并用 A 中的抽象机 $\mathcal{A}$ 来实现所需系统：因此，将 $\mathcal{A}$ 定义的设计相对于 $\varphi_\wp$ 所陈述的需求的正确性形式化为$L(\mathcal{A}) \subseteq L(\varphi_\wp)$，即机器实现的所有行为也都满足这些需求。有限状态自动机和正则语言的单子二阶逻辑正是这种情况。

遗憾的是，已知的理论下界表明，上述包含问题的判定是 PSPACE 完全的，因此一般而言难以处理。然而，近年来模型检测 [3], 取得了显著成功，产生了许多更精细的结果，解释了实用工具如何以及何时能够给出“可接受的复杂度”结果——尽管“可接受”一词取决于具体情境，因为在某些情况下，数小时甚至数周的运行时间也可能被视为可接受。在

简而言之，通常——粗略地说——我们以所采用逻辑较低的表达能力为代价，换取一种“仅为指数级”的复杂度（相对于逻辑公式的大小而言）；而单子二阶逻辑的最坏情况下复杂度甚至可能是非初等函数 [4].[4] 无论如何，这些讲义关注的并非复杂度问题，而是自动机识别器与单子二阶逻辑之间的等价性；这种等价性使上述基本包含问题具有可判定性。

参考文献

1. J. R. B¨uchi。《弱二阶算术与有限自动机》。数学逻辑季刊，6(1-6):66–92，1960。2. C. C. Elgot。《有限自动机设计及相关算术的判定问题》。美国数学会汇刊，98(1):21–52，1961。3. E. A. Emerson。《时序逻辑与模态逻辑》。载于《理论计算机科学手册》，*B* 卷：形式模型与语义（*B*），995–1072 页。1990。4. M. Frick 和 M. Grohe。《一阶逻辑与单子二阶逻辑的复杂性再探》。纯粹与应用逻辑年刊，130(1-3):3–31，2004。5. J. Henriksen、J. Jensen、M. Jørgensen、N. Klarlund、B. Paige、T. Rauhe 和 A. Sandholm。《Mona：实践中的单子二阶逻辑》。载于《系统构造与分析工具和算法》，第一届国际研讨会，*TACAS ’95*，《计算机科学讲义》*1019*，1995。6. C. A. R. Hoare。《计算机编程的公理基础》。*ACM* 通讯，12(10):576–580，1969。7. R. McNaughton 和 S. Papert。《无计数器自动机》。研究专著。麻省理工学院出版社，1971。8. W. Thomas。《理论计算机科学手册》（B 卷）。章节：无限对象上的自动机，133–191 页。麻省理工学院出版社，美国马萨诸塞州剑桥，1990。9. B. A. Trakhtenbrot。《有限自动机与单子谓词逻辑》（俄文）。《苏联科学院院刊》，140:326–329，1961。

[4] 不过，有一些工具在至少某些用单子二阶逻辑表达的性质情况下运行良好，这些案例相当值得注意 [5]。