arXiv:2304.03639v1 [cs.LG] 7 Apr 2023

线性循环网络在长序列括号计数中的理论条件与实证失效

纳丁·埃尔-纳贾尔 普拉纳瓦·马迪亚斯塔 蒂尔曼·韦德伦敦城市大学，英 **de** 国
{nadine.el-naggar,pranava.madhyastha,t.e.weyde}@city.ac.uk

摘要

先前研究已证实，具有无界激活函数的循环神经网络具备精确计数的能力。然而，研究也表明，循环神经网络难以有效训练，通常无法学会精确的计数行为。本文聚焦于这一问题，研究可能最简单的循环神经网络：线性单单元网络。我们对线性循环神经网络进行理论分析，找出模型表现出精确 计数行为的条件，并通过形式化证明说明这些条件既是必要的，也是充分的。我们还在两种不同设置下，围绕类似 Dyck-1 的平衡括号语言任务开展实证分析。我们观察到，采用标准方法训练时，线性循环神经网络通常不满足计数行为所需的充要条件。我们进一步探究训练序列长度和不同目标类别的选择如何影响模型的训练行为，以及线性循环神经网络模型有效逼近指示条件的能力。

1 引言

循环神经网络（RNNs）长期以来一直用于解决涉及序列数据的各种任务。它们曾是自然语言处理领域最常见的选择，但近年来已在很大程度上被 Transformer 取代。然而，最近人们对循环神经网络的理论层面重新产生了兴趣，相关研究包括Merrill 等人 (2020年)。另一项研究发现，采用压缩型和非压缩型（即无界）激活函数的循环神经网络在计数能力方面存在定性差异（Weiss 等，2018年）。这一发现与El-Naggar 等(2022年的研究结果相结合，表明即使采用无界激活函数的循环神经网络，也难以在超长序列上学会准确计数。因此，至关重要的是理解为什么循环神经网络尽管具备这种能力，却常常无法在实践中准确计数。

在本研究中，我们考察最简单形式的循环神经网络：线性单单元循环神经网络。我们的目标是：a）从理论上确定线性循环神经网络具备计数能力所需的条件；b）探究这些条件与训练后的线性循环神经网络模型的实证行为之间的关系。本文的主要贡献包括：a）确定了表明线性循环神经网络具有计数行为的两个条件；b）证明这些指标条件对于在线性循环神经网络中实现精确计数行为而言是充分且必要的；c）随后通过实证表明，线性循环神经网络通常无法学会精确计数，也不满足这些指标条件；最后，d）通过实证展示训练序列长度与指标值分布之间的关系。

2 相关工作

深度学习的成功激发了人们对理解神经网络理论性质的研究兴趣。早已知晓，循环神经网络是图灵完备的（Siegelmann andSontag, 1992年）。然而，Weiss 等人（2018年）表明，在使用有限精度状态时，不同循环神经网络架构在计数能力方面存在不同类别。循环神经网络与自动机及形式语言之间的关系已由 Merrill（2019年）进行研究，并且Merrill 等人 （2020年）构建了计数器机器的形式层级。这类分析通常基于对网络进行“饱和”处理，即用阶跃函数替代 Sigmoid 函数，使神经激活值变为二元值，从而简化分析。

然而在实践中，神经网络中的激活值种类繁多，计数甚至句法分析等系统性行为往往并未出现；这一问题在

三十多年来 Fodor 和 Pylyshyn （1988）一直在讨论这一问题。近期，Lake 和 Baroni （2017）指出了系统性行为方面的具体缺失，并设计了 SCAN，这是一项标准循环神经网络无法完成的合成语言处理任务。处理语言层次结构的一种传统方法是使用栈。研究者已提出多种神经栈版本，例如 Joulin 和 Mikolov （2015）、Grefenstette 等人 （2015）以及Chen 等人 （2020年），这些方法在 SCAN 上表现良好。Suzgun 等人 （2019b）使用栈增强型循环神经网络学习 Dyck-2 语言。Mali 等人（ 2021）开发了使神经栈行为更加稳定的方法，在长达 160 个词元的序列上取得了良好但并不完美的表现。

形式语言任务在超长序列上的泛化鲜少受到关注，因为这要求神经网络具有精确或近乎精确的行为，以避免误差不断累积，例如在计数时。Suzgun 等 (2019 年 a) 声称 LSTM 能够学会计数，但没有测试长度超过 100 的序列。Weiss 等人(2018年)报告称，通常难以训练的 ReLU 循环神经网络，以及在计数任务上更容易训练的 LSTM，在长度达数百个词元的序列上都表现失败。同样，在我们此前的工作中（ El-Naggar 等，2022年），我们表明，几乎所有 ReLU 循环神经网络和 LSTM 在长度为 1000 的序列上都会失败。

这些研究普遍表明，循环神经网络无法达到能够实现精确计数的配置。循环神经网络实现精确计数所需的一般条件尚不明确，而弄清这些条件对于深入理解循环神经网络的行为至关重要。本文通过研究线性单单元循环神经网络的情形，开始探讨这一问题。

3 线性循环神经网络中的计数行为

本节将正式定义平衡括号语言、平衡括号计数器和线性循环网络，并确定网络权重满足何种条件时，线性循环网络会表现为平衡括号计数器。我们的计数器定义基于Merrill (2020年) 所定义的通用计数器机（GCM）；为方便起见，我们也在附录A中列出了该定义。

GCM由词汇表 Σ、有限状态集 Q、初始状态 q_0、计数器更新函数 u、状态更新函数 δ 和接受掩码 F 定义。某些组件（例如状态）可以为空。计数器计算还使用零检验函数。输入字符串 x 由计数器一次处理一个令牌 x_t。计数器更新函数 u 用于以整数增量（$+m$）更新计数器值$\mathbf{c}$。计数器更新取决于当前输入令牌、当前状态以及当前状态的有限掩码。在我们的设定中，如果整个序列处理完毕后$\mathbf{c} = 0$，则称计数器机器接受一个序列。若计数器机器 M 接受语言 L，则称其 M接受 $s \iff s \in L$。

我们关注只包含一种括号的序列： Σ ={‘(’, ‘)’}。

定义 1.（平衡括号语言 BB）平衡括号语言 BB 定义如下：

$$BB = \{s \in \Sigma^* | \text{ count}(\text{ ‘(’}, s) = \text{count}(\text{ ‘)’}, s)\}.$$

对于该 BB 语言，左括号和右括号的排列顺序并不重要，重要的是序列中左括号和右括号的总数相等。Dyck-1 序列是 BB 序列的一个特例，其中右括号的数量在序列的任何位置都不会超过左括号的数量，也就是说，对于所有前缀都是如此。

我们关注单单元线性循环神经网络的计数能力。这些网络无法从所有可能序列构成的全集中接受 Dyck-1 序列，因为它们需要以不同方式处理负计数和正激活值，才能正确区分括号顺序正确与错误的序列。然而，单个线性神经元无法做到这一点，因此要从所有可能序列的全集中完整接受 Dyck-1 语言，还需要额外的机制。

以 Suzgun 等人(2019年a) 等人的先前研究为例，他们训练单单元循环神经网络模型学习 Dyck-1 语言时，数据集中只使用有效的 Dyck-1 序列，其中序列的任何位置都不会出现多余的右括号。不过，这种做法似乎不必要地限制了范围。因此，我们采用这种 BB 可由单单元线性循环神经网络完整接受的语言。

定义 2。（平衡括号计数器）
一般计数器机器是平衡括号计数器，当且仅当它接受 BB。

定义 3。（线性循环网络）线性循环网络（ LRN）是一种网络

其接收输入 x_t ，在每个时间步 t，该输入与前一时间步的激活 h_{t-1} 以及权重 W、U和 W_b一起用于生成激活 h_t，随后通过更新函数传递到下一时间步：

$$h_t = Wx_t + Uh_{t-1} + W_b.$$

这里， x_t 是一个独热编码的输入标记，如果我们将零检查 z 函数应用于 h_t 在处理最后一个输入后，基于 LRN 的计数器与 Merrill (2020年) 的定义存在以下偏差：

- 计数器值**c** 对应于 h_t（它是唯一从一个时间步传播到下一个时间步的值），它是实数而非整数，
- 更新函数（$+m$ 在计数器机器中）的结果是实数，具体而言：

$$a = Wx_t + W_b \text{ if } x_t = \text{‘(’ , and}$$
$$b = Wx_t + W_b \text{ if } x_t = \text{‘)’ .}$$

我们使用单单元 LRN 进行括号计数。因此， W 是一个与 x_t 维度相同的向量，而 U 和 W_b 是标量， m、**c**和 h_t也都是标量。

在定理 1中，我们将 LRN 的平衡括号计数器行为与其 权重的特定条件联系起来。我们定义了两个指示条件，并证明它们对于在LRN 中实现精确的计数行为而言是必要且充分的。

定理1。（线性循环网络计数指标）以下两个指标条件是线性循环网络接受平衡括号语言 BB的充要条件。

1. $\frac{a}{b} = -1$ (AB ratio)
2. $U = 1$ (recurrent weight).

证明1。 我们证明定理1 中的计数指标条件是线性循环网络接受平衡括号语言的充要条件。我们首先证明这些条件是必要的（第1部分），然后证明它们是充分的（第2部分）。

第 1 部分：我们证明，如果线性循环网络接受平衡括号语言，那么通过使用不同的输入序列（表1 ），定理 1中的计数指示条件成立，并据此推导出这些指示条件。如果线性循环网络接受平衡

案例输入		输出（h）	发现
1	‘(’	$h_1 \neq 0$	$a \neq 0$
2	‘)’	$h_1 \neq 0$	$b \neq 0$
3	‘()’	$h_2 = 0$	$b = -Ua$ 和 $U \neq 0$
4	‘((’	$h_2 \neq 0$	$U \neq -1$
5	‘(())’	$h_4 = 0$	$b + Ub + U^2a + U^3a = 0$
6	‘()()’	$h_4 = 0$	$b + Ua + U^2b + U^3a = 0$

表1：用于从定理中推导指示条件的输入序列1。

括号语言，那么当左括号和右括号数量相等时，输出激活为$h_t = 0$；否则为 $h_t \neq 0$。这等价于零检查函数 $z(h_t)$ 的结果为 0 或 1。因此，接下来的推导中我们不再纳入零检查函数。

情况1: $seq =$‘(’ $h_0 = 0, h_1 \neq 0$
$h_1 = a + Uh_0 = a \because a \neq 0$

情况2: $seq =$‘)’ $h_0 = 0, h_1 \neq 0$
$h_1 = b + Uh_0 = b \because b \neq 0$

情况3： $seq =$ ‘()’， $h_2 = 0$
$h_2 = b + Uab + Ua = 0$由情况1、2可得： $a \neq 0$和 $b \neq 0$
$\therefore b = -Ua$，以及 $U \neq 0$

情况 4： $seq =$‘((’, $h_2 \neq 0$
$h_2 = a + Ua$
$a + Ua \neq 0$
$\therefore U \neq -1$

情况 5： $seq =$‘(())’, $h_4 = 0$
$h_3 = b + Uh_2 = b + U(a + Ua)$
$h_4 = b + Uh_3 = b + U(b + U(a + Ua))$
$\therefore h_4 = b + Ub + U^2a + U^3a = 0$

情况 6： $seq =$‘()()’, $h_4 = 0$
$h_3 = a + Uh_2 = a + U(b + Ua)$
$h_4 = b + Uh_3 = b + U(a + U(b + Ua))$
$\therefore h_4 = b + Ua + U^2b + U^3a = 0$

综合情况 5 和 6 的发现。
$b + Ub + U^2a + U^3a = b + Ua + U^2b + U^3a$

两边同时减去 $b + U^3a$ ，再同时除以 U
$b + Ua = a + Ub$

重新排列并因式分解
$b - a = U(b - a)$
因此，我们得到 2 种可能的情况：

(a) $U = 1$，这意味着 $a = -b$，根据情况3
(b) $a = b$，根据情况1和2，我们知道$a \neq 0$ 和 $b \neq 0$，而 $U = -1$ 由情况3得出，这与情况4矛盾

$\therefore U = 1$ 并且 $a = -b$，也就是说，定理 1 中列出的计数指示条件成立，只要线性循环网络接受平衡括号语言。

第二部分：我们通过归纳法证明：如果定理 1 中列出的计数指示条件成立，那么线性循环网络就接受平衡括号语言。

每个序列都由 n 个左括号和 m 个右括号组成，输入词元 x_k 要么是“(”，要么是“）”。

基础情形： $k = 1$

- $x_1 =$‘(’, $n = 1$ and $m = 0$:
  $h_1 = a + Uh_0$
  $h_1 = a$
- $x_1 =$‘)’, $n = 0$ and $m = 1$:
  $h_1 = -a + Uh_0$
  $h_1 = -a$

对于 n 个左括号和 m 个右括号，下列等式满足基础情形，因此我们将其作为归纳假设:

$$h_k = (n - m) \times a$$

我们假设，对于长度为 k、由 n 个左括号和 m 个右括号组成的序列，该结论成立。我们通过归纳法证明：如果该结论对长度为 k 个词元的序列成立，那么对长度为 $k + 1$ 个词元的序列也成立。在归纳步骤中，我们各使用一次 $x_{k+1} =$“(”和 $x_{k+1} =$“）”。

归纳步骤：

- If $x_{k+1} =$‘(’:
  $h_{k+1} = ((n + 1) - m) \times a$
- If $x_{k+1} =$‘)’:
  $h_{k+1} = (n - (m + 1)) \times a$

根据前提，我们可以推导出:
$h_k = a+Uh_{k-1}$ 如果 $x_k =$‘(’，并且
$h_k = -a+Uh_{k-1}$如果 $x_k =$‘)’

- If $x_{k+1} =$‘(’:
  $h_{k+1} = a + h_k$
  Substitute $h_k = (n - m) \times a$
  $h_{k+1} = a + ((n - m) \times a)$
  $\therefore h_{k+1} = ((n + 1) - m) \times a$
- If $x_{k+1} =$‘)’:
  $h_{k+1} = -a + h_k$
  Substitute $h_k = (n - m) \times a$
  $h_{k+1} = -a + ((n - m) \times a)$
  $\therefore h_{k+1} = (n - (m + 1)) \times a$

因此，我们证明，如果定理中列出的计数指示条件1 在 LinearRecurrentNetwork 中得到满足，那么它就能接受平衡括号语言。 □

4 线性循环神经网络中的计数实践

我们开展实验，以分析这些模型，以及它们在训练中是否满足上一节所定义的条件。我们使用 2 个分类任务来评估我们的模型。

4.1 任务1：二分类

我们使用与先前工作相同的任务和模型（El-Naggar 等，2022年），即一个不带偏置、仅含一个输出神经元且采用 sigmoid 激活函数的线性 RNN，用于将序列的括号差分类为 > 0 或 ≤ 0（二元分类）。没有可训练偏置会减少模型的自由度，相当于将偏置（W_b）固定为 0，从而简化学习任务。我们也使用带有可训练偏置的相同模型。模型在长度为 2、4 和 8 个词元的序列上训练 100 个周期，共进行 10 次运行。每个序列的初始计数值（h_0）均为 0。我们检查模型的权重，并在图1中绘制每种训练集大小下训练所得模型的指示值（a/b，U）分布。我们观察到，模型并未满足指示条件，但随着训练序列长度增加，它们逐渐接近这些条件。我们还观察到，a/b 指示量的分布均值高于 -1，但训练序列越长，这种情况越不明显。

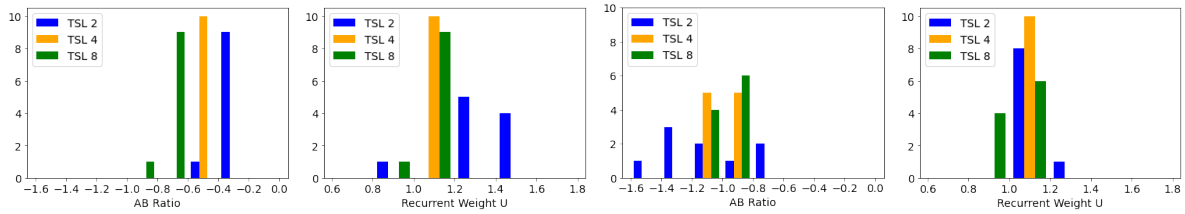
4.2 任务2：三元分类

我们还应用三元分类：> 0、$= 0$或 < 0。我们使用与任务 1 相同的模型，但不再使用带有 sigmoid 输出激活的单个输出神经元，而是使用 3 个输出神经元和带偏置的 softmax 输出层，这是能够完成此任务的最小配置。我们还使用带有可训练偏置的相同模型。

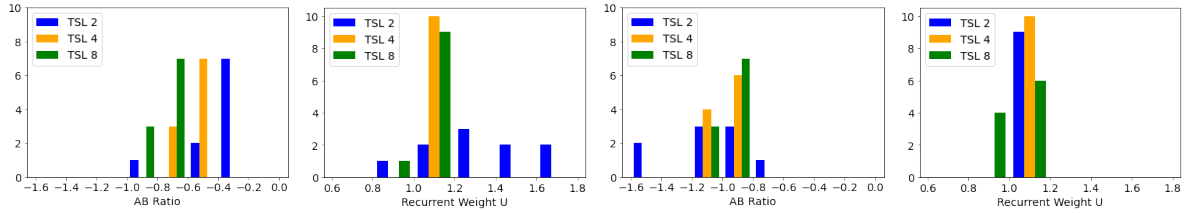
初始计数值（h_0）对每个序列而言均为 0，模型的训练方式与任务 1 中的模型相同。三元分类准确率略低，正如在类别更多时所预期的那样

分类 实验	训练 长度	训练 平均值（最小值/最大值）	20 个词元 平均值（最小值/最大值）	50 个词元 平均值（最小值/最大值）
二元（不含偏置）	2	100 (100/100)	69.2 (6.04/77.3)	69.0 (66.7/72.7)
二元（不含偏置）	4	100 (100/100)	94.8 (94.7/95.3)	89.3 (88.7/90.0)
二元（不含偏置）	8	100（100/100）	96.9（94.0/100）	92.7（78.7/98.0）
无偏三元	2	90（33.3/100）	55.6 (33.3/64.4)	51.4 (33.3/60.0)
无偏三元	4	100 (100/100)	79.5 (65.8/94.7)	67.2 (66.7/68.0)
无偏三元	8	100（100/100）	94.4（67.1/100）	85.7（66.7/100）
二元（含偏置）	2	100（100/100）	73.4 (63.3/100)	72.4（60.0/93.3）
二元（带偏置）	4	100 (100/100)	95.3（92.7/98.0）	86.0 (77.3/90.7)
二元（带偏置）	8	100（100/100）	95.2（85.3/100）	87.9（70.0/98.0）
三元（带偏置）	2	88.3（66.7/100）	58.0（38.2/67.6）	54.4（43.6/67.5）
三元（含偏置）	4	97.9 (79.2/100)	81.5 (64.4/100)	68.0 (65.3/73.3)
三元（带偏置）	8	100（100/100）	95.9 (83.6/100)	76.5 (65.3/100)

表 2：我们之前不带偏置的二分类实验的准确率指标（El-Naggar 等，2022年）、不带偏置的三分类实验，以及带偏置的二分类和三分类实验。



(a) AB比值二元（无偏置） (b)U值二元（无偏置） (c) AB比值三元（无偏置） (d) U值三元（无偏置）



(e) AB比值二元（偏置） (f) U值二元（偏置） (g) AB比值三元（偏置） (h) U值 三元 (偏置)

图1：在不同训练序列长度（TSL）下，无偏置（上）和有偏置（下）的二元和三元分类训练后指标。

对于较短的训练序列，这可能与相对于数据点而言模型参数数量更多有关。随着训练序列变长，准确率会提高。三元分类的平均值并未显示出 a/b 高于 -1的结果，如图 1所示。

5 结论与未来工作

尽管线性循环神经网络在理论上具备计数能力，但以往研究表明，这些模型往往难以有效地将计数行为泛化到长序列上。本研究提出了一组用于判定线性循环神经网络是否具有计数行为的充要条件，并证明满足这些条件等价于具有计数行为。为探究模型满足这些条件的程度，我们考察了在不同长度序列上训练、用于分类任务的模型参数。我们采用了二元和三元分类任务，发现模型并未完全满足这些条件，但确实有所趋近，而且随着序列长度增加，满足程度也越来越高。

根据这些发现，未来还有若干值得探索的方向。一种可能的研究方向是尽可能将这种方法扩展到 ReLU 循环神经网络和 LSTM。另一种选择是设计方法，确保在训练期间满足我们已确定的指示条件，从而提高模型的泛化能力。

参考文献

Xinyun Chen、Chen Liang、Adams Wei Yu、Dawn Song 和 Denny Zhou。2020年。通过神经符号栈式机器实现组合泛化。*arXiv* 预印本 *arXiv:2008.06662*。

Nadine El-Naggar、Pranava Madhyastha 和 Tillman Weyde。2022年。使用循环神经网络学习 Dyck-1 语言的实验。载于第三届类人计算研讨会（*HLC 2022*）论文集，与第二届国际联合学习与推理会议（*IJ-CLR 2022*）同期举行，英国温莎，*2022*年*9*月*28*日至*30*日，第3227卷*CEUR* 研讨会论文集，第24–28页。CEUR-WS.org。

Nadine El-Naggar、Pranava Madhyastha 和 Tillman Weyde。2022年。探索循环神经网络中计数行为的长期泛化能力。载于我简直不敢相信这不是更好的研讨会：通过实证证伪理解深度学习。

Jerry A Fodor 和 Zenon W Pylyshyn。1988年。联结主义与认知架构：批判性分析。认知，28(1-2):3–71。

Edward Grefenstette、Karl Moritz Hermann、Mustafa Suleyman 和 Phil Blunsom。2015。学习使用无界记忆进行转导。载于神经信息处理系统进展，第 1828–1836 页。

Armand Joulin 和 Tomas Mikolov。2015。利用栈增强循环网络推断算法模式。神经信息处理系统进展，第 28 卷。

Brenden M Lake 和 Marco Baroni。2017。无系统性的一般化：论序列到序列循环网络的组合能力。*arXiv* 预印本 *arXiv:1711.00350*。

Ankur Mali、Alexander Ororbia、Daniel Kifer 和 Lee Giles。2021。探究循环神经网络动态计数时反向传播的替代方法。载于国际语法归纳会议，第 154–175 页。PMLR。

William Merrill。2019年。作为自动机的序列神经网络。载于 深度学习与形式语言研讨会论文集：架起桥梁，第 1–13 页。

William Merrill。2020年。论实时计数器自动机的语言能力。*CoRR*，abs/2004.06866。

William Merrill、Gail Weiss、Yoav Goldberg、Roy Schwartz、Noah ASmith和 Eran Yahav。2020年。循环神经网络架构的形式层级。在 计算语言学协会第*58*届年会论文集，第443–459页。

Hava T. Siegelmann 和 Eduardo D. Sontag。1992年。论神经网络的计算能力。载于第五届计算学习年度研讨会论文集理论，COLT '92，第 440–449 页，美国纽约。计算机协会。

MiracSuzgun、Sebastian Gehrmann、YonatanBelinkov 和 StuartMShieber。2019年a。Lstm网络能够执行动态计数。*arXiv* 预印本 *arXiv:1906.03648*。

MiracSuzgun, Sebastian Gehrmann, YonatanBelinkov,and Stuart M Shieber. 2019年b. 记忆增强型循环神经网络能够学习广义 Dyck 语言。*arXiv* 预印本 *arXiv:1911.03329*。

Gail Weiss、Yoav Goldberg 和 Eran Yahav。2018年。关于用于语言识别的有限精度循环神经网络的实际计算能力。载于 第*56*届计算语言学协会年会论文集，*ACL 2018*年，澳大利亚墨尔本，*2018*年*7*月*15*日至*20*日，第*2*卷：短篇论文，第740–745页。计算语言学协会。

一种通用计数器机器

此定义引自 Merrill(2020年)。

定义 4。（通用计数器机器）一个 k 计数器是一个带有以下内容的元组 $\langle \Sigma, Q, q_0, u, \delta, F \rangle$：

1. 有限字母表 Σ
2. 一个有限状态集合 Q
3. 初始状态 q_0
4. 计数器更新函数

$$u : \Sigma \times Q \times \{0,1\}^k \to (\{+m : m \in \mathbb{Z}\} \cup \{\times 0\})^k$$

5. 状态转移函数

$$\delta : \Sigma \times Q \times \{0,1\}^k \to Q$$

6. 接受掩码

$$F \subseteq Q \times \{0,1\}$$

计数器机器计算在定义5中形式化。当前状态的有限掩码使用零检查函数 $z(\mathbf{v})$ 为向量 v创建，其中：

$$z(\mathbf{v})_i = \begin{cases} 0, & \text{if } v_i = 0 \\ 1, & \text{otherwise} \end{cases} \tag{1}$$

定义5. (计数器机器计算)令$\langle q,\mathbf{c}\rangle \in Q \times \mathbb{Z}^k$ 为机器M的配置。在读取输入 $x_t \in \Sigma$后，我们定义转移

$$\langle q, \mathbf{c} \rangle \to_{x_t} \langle \delta(x_t, q, z(\mathbf{c})), u(x_t, q, z(\mathbf{c}))(\mathbf{c}) \rangle$$