乘法线性逻辑量子算法

Lorenzo Saraiva[1], **Edward Hermann Haeusler**[1], **Vaston Costa** [2]

[1]里约热内卢天主教大学。R. Marquˆes de S˜ao Vicente, 225 - G´avea，里约热内卢 - RJ，巴西

[2]卡塔朗联邦大学。Av. Dr. Lamartine Pinto de Avelar 1120。Catal˜ao - GO，巴西

lorenzo.saraiva@hotmail.com, hermann@inf.puc-rio.br, vaston@ufcat.edu.br

arXiv:2302.09169v1 [cs.LO] 17 Feb 2023

摘要。 本文描述了一种量子算法，利用 *Grover* 搜索算法在线性逻辑一个子集的序列演算中进行证明搜索。我们简要概述了 *Grover* 搜索算法和线性逻辑，展示了该算法的详细步骤，随后给出了在量子模拟器上获得的结果。

1. 引言

量子计算为我们提供了时间复杂度优于任何经典对应算法的算法，其中之一便是用于在无序数据库中搜索元素的格罗弗搜索算法（GSA）[Grover 1997] 。GSA 被应用于多个领域，包括布尔可满足性问题、K均值、遗传算法和像素识别。在本文中，我们使用 GSA 来辅助搜索乘法线性逻辑子集的证明，以相比经典算法降低复杂度。我们展示了从线性逻辑序列到最终结果的量子电路构造，并给出结论。

2. 背景

GSA 是最著名的量子算法之一，其目标是在无序数据库中搜索一个元素。假设数据库包含 n 量子比特，并且其中的叠加态包含$N = 2^n$ 个元素，则其时间复杂度为 $\sqrt{N}$，优于任何经典算法。格罗弗算法在 n个量子比特上的主迭代通常包括以下步骤:

- 数据库初始化——在这一步中，对数据库量子比特施加算子 A ，使其从初始状态 $|0\rangle^{\otimes n}$ 变为目标态 $|\Psi\rangle$。该状态通常是均匀叠加态，并且 $A = H^{\otimes n}$。

- 预言机调用——在这一步中，对制备态 $|\Psi\rangle$施加预言机 O 。预言机将翻转搜索值 x_t 的相位，使得: $O|x_s \neq x_t\rangle = |x_s\rangle O|x_s = x_t\rangle = |-x_s\rangle$

- 振幅放大 - 在这一步中，使用一个算子 D 来放大预言机标记的状态的振幅。为此，会执行关于平均值反演（IAM）。

一般而言，Grover 迭代可描述为:

$$G = AOA^T D$$

为了最大限度地提高测量到目标态的概率，Grover 迭代必须重复 $\lfloor \pi\sqrt{N}/4 \rfloor$ 次。我们的工作沿用了 Alsing 和 McDonald 2011] 使用 GSA 的 Alsing 的纠缠数据库搜索[。Alsing 算法的主要特点是，它不使用 $A = H^{\otimes n}$ 来制备均匀叠加态，而是选择 A 来编码任意的键值对列表 $\{s, t\}$。因此，算法的输入是一个包含两侧的纠缠数据库，每一侧各包含键值对的一部分。左侧的每个条目都与右侧的一个条目纠缠。在 GSA 中，必须知道搜索值才能构造预言机。而使用 Alsing 算法时，则可以根据左侧一个已知条目 s_1 构造预言机，应用 GSA，然后测量右侧，从而恢复与 s_1纠缠的未知值 t_1 。

3. 问题描述

线性逻辑是经典逻辑和直觉主义逻辑的一种扩展，强调公式作为资源的作用。因此，它不允许收缩和削弱规则应用于所有公式，而只允许应用于带有特殊标记的公式[Di Cosmo 和 Miller 2019]。由于处理这些标记的逻辑规则与 S4 等系统中的模态在（形式上）相似，这些标记可以被视为模态。没有收缩和削弱规则，使得线性逻辑具有两种不同版本的合取和析取：加法版本和乘法版本。例如，经典逻辑中的 $\wedge$ （与）被分为加法版本 &（with）和乘法版本 $\otimes$ （张量）。线性逻辑也有一个序列演算证明系统。在此背景下，在仅含乘法运算的线性逻辑版本中寻找无割证明的算法，其最坏情况下的时间复杂度为 2^k，其中 k 是原子公式的数量。直觉主义线性逻辑中仅涉及乘法联结词的子集称为（直觉主义）乘法线性逻辑（IMLL）。本文将使用 IMLL 的一个子集 IMLL-$\otimes$，其中只使用张量联结词。

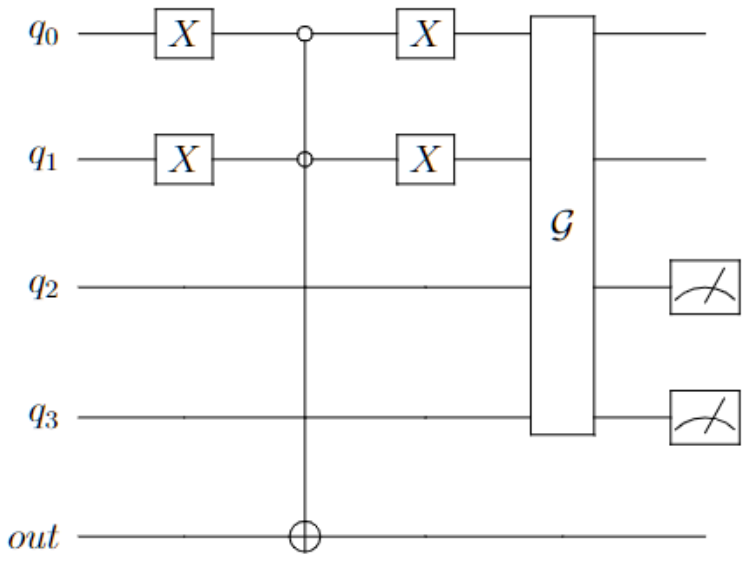

图 1. 用于 k=2 and 搜索与 0 关联值的预言机电路

考虑一个含有 $k = 4$ 个原子子句的线性逻辑序列

$$A \otimes (B \otimes (C \otimes D)) \vdash D \otimes (B \otimes (A \otimes C)).$$

我们希望找到一系列划分，以验证这是一个有效证明。我们可以应用两条规则： $\otimes$-左规则和 $\otimes$-右规则。在经典算法中，我们依次应用这些划分，直到得到一个有效公理。

- 应用一条可能的规则，直到只剩下左公理，
- 如果所有公理都有效，则该序列有效；否则重新开始。

由于有 2^k 种可能的划分，该算法的时间复杂度为 $O(2^k)$，其中 k 是原子子句的数量。

算法1概述

要求： k 份包含 $2n$ 个量子比特和 k 对的数据库，其中 $n = \lceil \log k \rceil$
$numIterations \leftarrow \lfloor (\pi \times \sqrt{N/4}) \rfloor$**循环** $i < N$ **循环** $j < numIterations$ **执行**构建预言机(n, target) 追加扩散算子(A) 测量()**结束循环 结束循环**

4. 求解步骤

我们的量子算法以一个由两侧组成的纠缠数据库作为输入，每侧都是一对中的一部分。左侧的每个条目都与右侧的一个条目纠缠在一起，并且两侧的条目都没有排序。我们将每一对中的左侧称为搜索部分，或 s，并将右侧称为目标，或 t。为了能够在 $\sqrt{k}$ 步内执行该算法，需要有k 份配对数据库副本，其中 k 是不同原子子句的数量。构建此数据库的复杂度不计入考量。我们的算法还展示了一种显式的动态构造方法，用于根据搜索值构造格罗弗预言机。

4.1. 准备纠缠数据库

在开始算法之前，必须构造一个能够准确表示序列的纠缠数据库。为此，我们需要 $k \times 2n$ 个量子比特。然后，将这些对$|a\rangle\,|b\rangle$ 编码为 $|Na + b\rangle$，其中 a和 b是序列每一侧中子句的位置。假设我们有 $k = 8$和 $n = 3$，其中 n 是表示包含 k 个值的解空间所需的量子比特数。因此，我们包含 8 个解的纠缠数据库将具有 2 组 n 个量子比特，每组分别表示 $8 = k = 2^n$ 个值。我们会将两组各含 3 个量子比特的组视为一个数组，并使用 k 个 k^2 种总可能性中的可能性，将所得编码制备为叠加态；这些可能性可存储在 $2n$ 个量子比特中。然后，我们需要 k 个寄存器副本，每个子句对应一个副本。数据库的构造过程未明确展示，但其复杂度为 $O(n)$ 或 $O(\log k)$[Alsing 和 McDonald 2011],

总计为 $O(k \log k)$。这一过程并不严格属于算法的一部分，因为算法只接收预先构造的纠缠数据库。[1]

4.2. $\otimes$-左规则

算法本身的第一步是应用 $\otimes$-左规则，直到其无法再被应用为止，因此我们得到如下形式的序列：

$$A^1, A^2, ..., A^N \vdash B \otimes \Delta$$

其中 $\Delta = A \otimes(\Delta)$ 或 $\Delta = A$ 现在，我们可以使用纠缠数据库来找出右侧最左边原子子句的正确划分。

4.3. Grover 搜索

现在我们已经拥有由 k 份 $2n$ 个量子比特组成的纠缠数据库，可以执行 GSA。我们首先选择右侧最左边的原子子句。第一步是为左侧选定元素动态构造预言机。该预言机的构造考虑了所选子句位置的二进制表示。我们施加必要的 X 门，使所有搜索空间量子比特处于 $|1\rangle$，并以一个预备量子比特为目标施加多重受控 Toffoli 门，以执行相位回踢，如 1 中所示。该预言机仅应用于搜索部分，即 $2n$ 个量子比特第一份拷贝中的前 n 个量子比特。随后，对全部 $2n$ 个量子比特应用 Grover 算子进行振幅放大，共 $\sqrt{k}$ 次，并对 $2n$ 个量子比特的右侧进行测量。

对于 $n = 2$ 个量子比特，这种电路的一个示例如 1 所示。需要注意的是，在 GSA 的 IAM 步骤中，必须应用 A 算子，这需要 $\log k$ 步，因此 GSA 步骤的总体复杂度为 $O(\sqrt{k} \log k)$。这一过程会找到一对中对应的条目，但我们需要找到 k 对对应的条目。问题在于，测量量子比特会破坏与这些条目对相对应的预备叠加态。因此，我们需要 k 份预备好的 $2n$ 量子比特纠缠数据库副本——于是我们对数据库的不同副本分别针对每一对执行 GSA，总共需要 $k^{1.5}$ 步，即 k 次 $\sqrt{k}$ 步。

5. 示例

我们想要确定

$$A \otimes (B \otimes (C \otimes D)) \vdash D \otimes (B \otimes (A \otimes C)))$$

是线性逻辑中的一个有效序列。我们有 $k = 4$，因此有 $n = \log_2^4 = 2$，故每一侧使用 2 个量子比特，并且总共每个副本使用 $2n$。我们将构造该序列的纠缠表示。

A	0	2	
B		1	1
C	2	3	
D	3	0	

[1] 在我们的情形中，对于纠缠态，必须存储用于编码副本的门序列 A of 纠缠数据库状态，以便稍后在 GSA 的 IAM 步骤中使用。

$$\frac{\Delta, B_0, B_1 \vdash \gamma}{\Delta, B_1 \otimes B_2 \vdash \gamma} \otimes\text{-Left} \qquad \frac{\Delta_0 \vdash A_0 \qquad \Delta_1 \vdash B_1}{\Delta_0, \Delta_1 \vdash A_0 \otimes A_1} \otimes\text{-Right} \qquad \frac{}{B \vdash B}$$

图 2. $\otimes$专属线性逻辑的规则

使用公式 $|ka + b\rangle$，其中 $k = 4$，我们有

A	k0 + 2 = 2
B	k1 + 1 = 5
C	k2 + 3 = 11
D	k3 + 0 = 12

因此，量子数据库的状态为

$$\sqrt{\tfrac{1}{4}}(|2\rangle + |5\rangle + |11\rangle + |12\rangle)$$

or

$$\sqrt{\tfrac{1}{4}}(|0010\rangle + |0101\rangle + |1011\rangle + |1100\rangle)$$

我们在任意一侧执行 Grover 搜索，并能够恢复另一侧的值。不过，先让我们回到序列。 $\otimes$ 的规则如图 2 所示。

由于 $\otimes$ 是一个二元运算符，我们无法搜索括号内的值并应用规则，因此我们将序列视为

$$A \otimes \Delta^1 \vdash D \otimes \Delta^2$$

因此，我们先连续应用 $\otimes$-左规则，直到每个原子子句都单独出现

$$\dfrac{\dfrac{\dfrac{A, B, C, D \vdash D \otimes \Delta^2}{A, B, C \otimes D \vdash D \otimes \Delta^2} \otimes\text{-Left}}{A, B \otimes \Delta^3 \vdash D \otimes \Delta^2} \otimes\text{-Left}}{A \otimes \Delta^1 \vdash D \otimes \Delta^2} \otimes\text{-Left}$$

现在，我们对右侧的每个条目运行量子算法，并按照出现顺序将结果应用到序列中。 D 被编码为一对 (3，0)，但算法只知道 0，也就是我们正在查询的值的位置，该位置由 $\sqrt{\frac{1}{4}}\,|1100\rangle$ 编码。我们将使用图示电路，在代表这对数中 0 部分的最后两个量子比特上应用预言机，然后测量前两个量子比特；结果很可能是 3。对每个

右侧子句都执行这一过程，因此现在我们只需按照索引 (3，1，0，2) 应用划分，结果如下

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{A \vdash A \quad C \vdash C}{A, C \vdash A \otimes C}\otimes\text{-Right} \quad B \vdash B}{A, B, C \vdash B \otimes (A \otimes C)}\otimes\text{-Right} \quad D \vdash D}{A, B, C, D \vdash D \otimes \Delta^2}\otimes\text{-Right}}{A, B, C \otimes D \vdash D \otimes \Delta^2}\otimes\text{-Left}}{A, B \otimes \Delta^3 \vdash D \otimes \Delta^2}\otimes\text{-Left}}{A \otimes \Delta^1 \vdash D \otimes \Delta^2}\otimes\text{-Left}$$

6. 结果

对于给定的纠缠数据库状态，我们的算法的时间复杂度为 $O(k^{1.5}\log k)$，空间量子比特复杂度为 $\log k$，其中 k 是序列中的原子子句数。即使将数据库的构建过程也考虑在内（该过程需要 $O(k\log\ k)$ 步），时间复杂度仍为 $O(k^{1.5}\log\ k + k\log k)$ $=$ $O\ k^{1.5}\quad k$ (log)，优于经典算法。

此外，值得注意的是，当 $k > 4$ 时，我们需要使用具有两个以上控制量子比特的受控非门。为此，我们需要串联托福利门的结果，这会引入额外的 $n-2$ 辅助量子比特[Piro 等人 2020]。我们在 IBM 提供的多个模拟器上运行了电路，例如 *QASM* 模拟器 和*MPS* 模拟器。每次执行都运行电路 1000 次。我们使用 $2 \times \log k \doteq 12$个量子比特作为搜索空间，以高精度测试了该算法的实现，最多处理 64 个原子子句。

7. 未来工作

尽管该方案利用 GSA 在搜索匹配对时获得了优势，但仍存在一些缺点。首先，每次执行都需要准备量子数据库，因为量子态会在过程中被破坏。另一个问题是，该算法并非完全量子化：虽然使用 GSA 找到了索引匹配，但划分是根据每个原子子句的位置以经典方式完成的，因此有人可能会认为这会增加复杂度开销。为此，目前正在开发一种不同的量子方法，其中每个量子比特的取值将表示某个特定划分中原子子句所在的一侧。

8. 全量子方法

该算法也使用 GSA 来协助对直觉主义乘法线性逻辑进行证明搜索，但它与第一个算法之间存在相当大的差异。现在，我们既不使用 Alsing 的纠缠数据库，也无需在执行前准备特定的量子态。该算法使用 $(k-1)\ + \log k$ 个量子比特，其中 k 是序列右侧原子子句的数量。前 $(k-1)$ 个量子比特表示在 $(k-1)$ 个划分中的每一个划分内由某个子句选取的一侧，后面的 $\log k$ 个量子比特则充当子句的索引。测得为 0 的量子比特表示某个子句将在一次划分中被分配到左侧，而 1 则表示其将被分配到右侧。先从最简单的情况开始：

$$A, B \vdash A \otimes B$$

对于 $k = 2$，我们将需要 $(k-1) + \log k = 2$ 个量子比特。表示正确划分的量子态为 $|00\rangle + |11\rangle$。$|00\rangle$ 状态表示 A 被分配到右侧，而 $|11\rangle$ 表示 B 被分配到左侧。GSA 预言机将把这两个状态标记为正确状态。这些状态由序列的右侧定义。让我们来看一个稍微复杂一些的示例：

$$A, B, C, D \vdash D \otimes (B \otimes (A \otimes C)))$$

我们将查看右侧，以定义将由预言机标记的状态。首先，D 将移至左侧，其余所有内容移至右侧。D 将不再进行后续划分，在这种情况下，我们会将其对应状态的其余部分填充为 0。因此，预言机的一个正确状态是 $|010|11\rangle$。采用类似的过程，我们可以构造出另外三个状态：$|110|00\rangle$、$|100|01\rangle$ 和 $|111|10\rangle$，分别对应 A、B 和 C。现在，我们只需对 GSA 应用足够多次测量这四种可能性，就能恢复形成有效序列所需的划分。其时间复杂度为 $\sqrt{\frac{2^{k+\log k}}{k}}$。这可以简化为：

$2^{k+\log k} = 2^k \times 2^{\log k} = 2^k \times k$

$\sqrt{\frac{2^k \times k}{k}} = \sqrt{2^k}$

这就是 GSA 所带来的预期二次加速。

9. 添加线性蕴含

遵循全量子方法，下一步是将线性蕴含加入算法所接受的连接词中。这会带来一些挑战。首先，我们不能再将右侧用作预言机依据 $\otimes$-Right 规则应用连续划分的固定参照——如果加入线性蕴含，现在原子子句可能会根据规则切换所在的一侧，并且初始序列不再需要在两侧具有数量平衡的原子子句。因此，我们不应只规定左侧的划分，使其遵循右侧的固定顺序，而是需要处理每一个原子子句。此外，在应用 $\multimap$-Left 规则时，我们有四个“去向”选项：左相继式的左侧、左相继式的右侧、右相继式的左侧以及右相继式的右侧。$\otimes$-Right 也存在同样的问题，因为现在我们必须明确说明每个子句将去往何处。为了解决这一问题，每一步将使用 2 个量子比特，而不是一个。量子比特对中的第一个表示子句将去往哪个相继式，0 表示左侧，1 表示右侧。第二个表示子句将去往相继式的哪一侧，同样地，0 表示左侧，1 表示右侧。应用 $\multimap$-Right 时，将视为所有子句都去往左相继式。下面来看一个简单的例子：

$$A^1, A^2 \multimap B^1 \vdash C^1 \multimap B^2, C^2$$
$$A^1, A^2 \multimap B^1, C^1 \vdash B^2, C^2$$

$$A^1 \vdash A^2 \qquad\qquad B^1, C^1 \vdash B^2, C^2$$

因此，预言机的正确状态将是：

$$A^1 = |0000|000\rangle$$
$$A^2 = |0001|001\rangle$$
$$B^1 = |0010|010\rangle$$
$$B^2 = |0111|011\rangle$$
$$C^1 = |0010|100\rangle$$
$$C^2 = |0111|101\rangle$$

这里有几点值得指出。首先是量子比特数量的增加。上一个解的复杂度为 $\sqrt{2^k}$，其中 $k = n/2$，而 n 是原子子句的总数。相比之下，这个解的复杂度为 $\sqrt{\binom{2^{c+\log n}}{n}}$。以类似的方式简化：

$2^{c+\log n} = 2^c \times 2^{\log n} = 2^c \times n$

$\sqrt{\frac{2^c \times c}{c}} = \sqrt{2^{2c}}$

2^c 是最终复杂度。

参考文献

Alsing, P. M. 和 McDonald, N.（2011）。使用纠缠数据库状态的格罗弗搜索算法。载于量子信息与计算 *IX*，第 8057 卷，页码 80570R。国际光学与光子学学会。

Di Cosmo, R. 和 Miller, D.（2019）。线性逻辑。载于 Zalta, E. N. 编，《斯坦福哲学百科全书》。形而上学研究实验室，斯坦福大学，2019年夏季版。Grover, L. K.（1997）。量子力学有助于在干草堆中寻找一根针。《物理评论快报》，79(2)：325。

Piro, F., Askarpour, M. 和 Di Nitto, E. (2020). 将恰好 1-SAT 求解器推广至任意数量的变量、子句和 k。载于第一届软件工程与技术国际研讨会（*Q-SET 2020*），第 2705 卷，第 27–37 页。CEUR-WS。