梯径:“修修补补”和“重复利用”如何增加复杂度与信息*

刘宇，北京师范大学珠海校区复杂系统国际科学中心†

摘要

信息和复杂性在诸多领域都是重要概念，比如分子生物学、进化理论和外太空生物学。这些量的一些度量（比如香农熵及相关的复杂性度量）是从统计集合的角度定义因而不能应用于单个客体；而诸如 Kolmogorov 复杂度的另一些度量则不容易或不能计算。我们这里提出“梯径”的概念用以描述一个客体如何利用重复单元，而解构成具有层级关系的结构。从梯径中能自然导出两个指标：梯径度和有序度，代表了复杂性的两个轴。我们阐述了梯径如何运用在字符串和图案上，并且试图说明所有能够进化的系统都可以用梯径描述。更进一步，我们讨论了其在人类语言和生命起源方面研究的潜在应用。梯径提供了一种全新的刻画蕴含在单一客体（或系统）中信息的方法，能够帮助我们理解系统的进化，特别是和生命起源相关的话题。

关键词

复杂性、句法信息 (Syntactic information)、香农熵、柯氏复杂度 (Kolmogorov complexity)、层级 (Hierarchy)、模块、生命起源、外星信号、语言、进化

目录

*翻译自 Yu Liu, Zengru Di, Philip Gerlee. Ladderpath Approach: How Tinkering and Reuse Increase Complexity and Information. *Entropy*, 24(8):1082, 2022. 并增加了一些脚注。

†2022.08.05. yu.ernest.liu@bnu.edu.cn

1 引言

在一段采访中，John Maynard Smith 说 19 世纪是能量的世纪，那时科学和工程都关心将能量从一种形式变成另一种（如在蒸汽机里从化学到机械力，在发电机里从机械力到电）；而 20 世纪是关于信息的，特别是信息的转化 [1]。这种变迁在生物学中尤为明显，作为遗传信息载体的 DNA 的发现和随后整个人类基因组的测序已经使得生物学变成了一门关于信息的科学。

尽管大多数人都对信息包含什么有一个直观的认识，但是严格地定义信息却似乎非常困难，部分是因为这个词语具有的不那么统一的含义。有时候信息指代关于世界的某种知识或事实（即语义信息 semantic information），有时候也可以指代某个客体的某种结构特征，而与其具体的意义和解读无关（即句法信息 syntactic information）[2]。

考虑以下句子：*Claude was an American mathematician, and Claude spent his childhood in Michigan.* 如果不能正确解读组成该句子的单词，那么该句子的语义信息是不可能被知道的。为知道语义信息，我们必须理解“mathematician”在真实世界中到底指代什么，但这种指代是不能从单独这个句子中推导出的[1]。所以，当考虑一个处于孤立状态的客体时，唯一合理的对其信息的量度就是其句法信息。

当考虑信号在通信信道中传输的时候（这也正是香农提出信息论的初衷），我们希望所传输的每一个符号都尽可能地富含信息。但是当我们观察语言或者生物的时候，我们会注意到，信息总是通过有冗余的结构而传达。

这些我们通常认为复杂的结构总是在完全有序（比如，一个晶体结构）和完全随机（比如，理想气体中的分子）之间。直觉来看，我们不希望赋予一个高度有序的序列很高的复杂度（比如，由很多重复单词构成的句子），但是同时，我们也不希望称一个完全随机的序列是复杂的（因为它没有任何结构）。然而，我们希望赋予处于重复和随机中间的那些序列很高的复杂度 [3, 4]。这个直觉上的要求却使得定义复杂度非常困难。

第一个在上述思路下尝试定量化复杂性的是 Kolmogorov [5] 和后来改进其理论的 Chaitin [6, 7, 8]。该类理论通常称为算法复杂度 (algorithmic complexity)，其将一个序列的复杂度定义为生成该序列的最短计算机程序的长度。这确实很有道理，但是鉴于通用图灵机的停机问题，这种定义的缺点是，对于任意序列，并没有普适的方法能计算该复杂度 [9, 10]。尽管如此，如果定义在弱化 (weaker class) 的图灵机上（比如在特定的内容或场景下），算法复杂度还是可以近似计算的。比如，广泛用在数据压缩中的著名的 Lempel-Ziv 算法 [11, 12, 13] 就已经接近最优的无损压缩率。另一个例子是“physical complexity”，它设计出来专门为刻画基因序列所编码的关于其周遭环境的信息量 [14]。

[1]举个例子，某些人工智能比如 OpenAI，虽然能写出流畅的完全没有语法错误的句子、文章，但却更像是那种天马行空、毫不考虑常识的“作家”（当然可以是很好的魔幻主义、浪漫主义作家），其中的原因就是 OpenAI 是通过大量的文章来训练得到的，它确实掌握了单词、短语（某些字符串总是重复出现，那么它们一定是单词短语）、甚至句子之间的关系等等，但是并不能跟客观的事物联系在一起。所以 OpenAI 掌握的是语言结构本身蕴含的信息，但是并不跟客观事物产生联系。这也就是为什么它很难从这些信息中提取出“常识”（除非强行灌输），因为常识是通过对客观事物的联系获得的，比如“石头很硬”（虽然这种知识并不一定是正确的）。通过这个例子也可以看出，语言的信息其实包括两层意思，第一层是文字、图案或声音等形式本身蕴含的句法结构上的信息（即那些重复的结构），第二层是跟客观事物、世界的联系，即语义信息。本文中，我们暂时先没有考虑第二层意思，只集中在第一层意思。

另一大类复杂度测量建立在香农的信息论之上，特别是香农熵，其定义为 $H = -\sum p_i \log_2 p_i$，其中 p_i 表示字符 i 出现在此序列中的概率，该处的求和是指对所有可能的字符求和 [15]（香农熵本身对测量复杂度并不是很有用，因为它对于完全随机的序列会取最大值）。比如，Grassberger 引入了一个称为“Effective Measure Complexity”的量，其定义为一个序列所蕴含的可用于预测下一个字符的平均信息量 [16]。另一个相似的量度是由 Crutchfield 和 Young 引入的“Statistical Complexity”，其衡量序列中所呈现出的因果状态 (causal states) 的概率分布的香农熵 [3]。统计学中，另一个相关概念是 Fisher information，其刻画一个随机变量分布相对于其参数的相对熵的曲率，也就可以衡量出该随机变量所蕴含的信息量，它可以用在包括衡量学习任务的复杂度 [17]、分析经济市场产生的信号 [18] 等诸多方面。

上文提到的 Kolmogorov 复杂度是关于单个客体的绝对信息 [19]，所以对具体序列的排列模式是敏感的，但香农熵及其相关的复杂度测量却是一个统计意义上的概念，它们只能定义在某些处于统计集合之中的序列或客体上（或等价地，无限长的序列），故对具体序列的排列模式不敏感。一种解决方案是假定序列中所有字符在默认情况下都是满足均匀分布的。然而，这种解决方案忽略了储存在一个特定序列中的信息（或“意义”）[2]。因此如何刻画有限长的序列的句法信息和复杂度仍然是一个开放问题。

由于信息是跟重复结构、重复模式联系在一起的（比如 Lempel-Ziv 算法即是利用了重复子序列来压缩信息 [20, 21]），上面的问题也可以从另外一个角度来看，从更普遍的观点上看。通过结合或改变现有客体来生成新客体（比如基因序列、分子、科技发明）的想法在 1977 年由 François Jacob 提出，称为“修修补补的进化 (evolution as tinkering)”[22]。这个关于信息的累积的想法已经为很多研究领域打下了基础，比如蛋白质相互作用 [23, 24]、软件模块化 [25, 26]，也仍然在持续地启发新的研究（从更普遍的角度），比如网络复杂度的演化，其中重用扮演了极其重要的角色 [27]。1997 年 Donald Knuth 在他著名的计算机科学的著作中详细研究了一个有类似想法的问题“addition chain”：整数 n 最短的 addition chain 即是最高效的构造 n 的加法序列（其中，已经被构造出来的整数可以被再次利用）[28]。不久前，“pathway complexity”[29] 和“assembly space”[30, 31] 的概念被提出，通过计数需要多少步骤构造出原本的客体（其中，已经被构造出来的结构可以在随后的步骤中被重复利用）来刻画该客体的复杂度。该方法已经被用在探测生命特征中，其中的假说是，如果一种足够复杂的分子能够被大量地发现，那么生命过程一定参与了其中 [32]。最近，“molecular assembly tree”的概念也被提出，用于刻画一组互不相同的分子间的层级关系，展示出在各个领域的巨大潜力，比如在新药设计和对生命起源的研究中 [33]。

类似于上述想法，即“evolution as tinkering”、“addition chain”、“assembly theory”，本文构建了另一方法来刻画信息和复杂度，着眼在生成给定客体（如序列、句子、图片、分子、蛋白质、建筑结构等）的过程。我们称其为“梯径”方法，它并不着眼在关于计算的抽象理论之上，比如 Kolmogorov 复杂度和香农熵，而是受以下过程的启发，即影响现实世界的修修补补的过程和重用的机制。

以下第 2节中，我们从零开始详细阐述“梯径”方法，依次给出相关定义，随后介绍计算最短梯径的算法。第 3中，我们将梯径与进化过程联系起来。最后在第 4节中，我们讨论如何解读未知信号，以及关于生命和进化的话题。

[2]上一个脚注提到，一句话会有两层意思，第一层是句法信息，第二层是语义信息。这里要强调的是，香农熵并不能描述语义信息。但是，由于我们（不管是物理学上还是信息学上还是任何其他学科）并不知道怎样定量地描述这个“语义信息”，又由于“香农熵”（或“信息熵”）有明确的公式来计算，所以很容易将这二者混淆起来：即认为语义信息就是等同于香农熵，即等同于一句话给我们的惊讶程度。确实，将惊讶程度等同于语义信息在直觉上似乎是讲得通的，因为如果某句话让我们感到非常意外，那说明它确实是包含了我们以前不知道的信息，相反，如果某句话完全不让我们意外，那么说明这句话并没有给我们新的信息。
然而，香农熵描述的这个惊讶程度跟直觉上的惊讶程度是不一样的。香农熵是让这个句子跟所有完全随机的句子比较，描述该句子到底有多大概率从随机中产生出来，该句子越随机那么它的香农熵越大；而直觉上的惊讶程度是跟我们已知的信息相比较，差别越大惊讶程度越大。这二者有本质区别：前者是跟一群句子（所有的随机句子）比较，后者是只跟一个句子比较（记忆里的某个特定句子）；前者是跟非特定的句子比较（将所有随机句子看成一个整体），后者是跟特定的句子比较（记忆里的某特定句子）。

2 梯径 (Ladderpath)：从零开始推导

2.1 一个理想实验

为了阐述定义梯径的必要性，我们做一个理想实验：想像我们接收到宇宙中某处传来的一段字符（假设已经翻译成英文字符的形式）：

ACXLGICXGOXEMZBRCNKXACXLPICXEMZBRCNKX

那么该字符串是否包含某种信息，如果是，我们该怎样理解它呢？首先，我们可能会说该字符串本身就包含着语义上的信息（即该信号的主人想向我们传达的信息），但我们先不这样考虑，因为我们目前并没有方法赋予该字符串中的字符以现实中的明确意义（即还没有方法解读）。

那对于从该字符串中抽取信息这个问题应该如何考虑呢？应该从搜索重复结构出发。这串字符里，EMZBRCNKX 出现了 2 次，而在这么短的字符串中，完全随机地出现这么长的相同的字符串 2 次，概率是非常低的（如果重复出现更多次，比如 3 次、4 次、100 次，那么我们就会更确定它不是随机的，则一定有某种明确的语义）。虽然随机地重复 2 次并不是完全不可能，但是在本文中我们先姑且认为只要重复出现 2 次及以上，就不是随机的。这种假设下，我们就可以说 EMZBRCNKX 一定是有明确语义的。尽管如此，至于 EMZBRCNKX 到底指的是什么，并不能通过这句话本身推测出来。如果我们从其他渠道看到 EMZBRCNKX 频繁地跟另外的一些文字、图画、客观物体等出现在一起，那么我们就可以推测出它具体的指代。但是现在，我们只能说 EMZBRCNKX 一定意味着什么，或者它一定是外星人语言中的一个单词或短语。另外，我们还可以看到，ACX 也出现了两次，所以它也一定意味着什么。

那对于那些没有重复出现的字符串，比如 LGIC、LPIC，它们是否也有明确的意义，或者说是否具有语义信息呢？如果仅凭这一个字符串的话，我们是永远不可能知道的。但是如果有其他渠道发现 LPIC 经常重复地出现，那么我们也可以确定 LPIC 一定有明确意义。所以，对于这一个孤立的字符串，其语义信息仅存在于 EMZBRCNKX 和 ACX 之中。[3]

2.2 定义“生成操作”(Generation-operation)

这一节我们为给出“梯径”的严格定义做准备。梯径的定义源于这样一个问题：假设我们有一个包含英文字符 A, B, C, D, E, F 的集合，现在需要获得一个特定的英文字符串

$$\mathcal{X} = \text{ABCDBCDBCDCDEFEF}$$

问如何以最快的方式获得 $\mathcal{X}$（即，需要的最少步数）？当然这个问题可以推广到普遍的情形，比如最初的集合是原子，需要获得的是一个特定的分子（在我们以前的文章 [33] 中，我们研究了关于分子的一个初始想法）；或者最初的集合是五金零部件，需要获得的是一辆自行车；或者最初的集合是各种食材、调料，需要获得的是一道特定的菜肴，等等。这里我们以字符串作为一个简单而不失普遍性的例子。

首先，我们定义一个“**基础集**”，记为 $\mathbf{S}_0 = \{\text{A(0), B(0), C(0), D(0), E(0), F(0)}\}$，它实际上是一种特殊的“偏序多重集”，即，其元素是偏序的（有序的部分用双斜杠“//”隔开，两个“//”之间称为“一层”；无序的部分用逗号“,”隔开），并且记录元素的个数（称为“重数”），写在元素后面的括号中。注意，$\mathbf{S}_0$ 需要根据具体问题而事先确定，然后才能进行分析（见第 4.1节中更多的讨论）。在本例中，我们将基础集定义为由单个字符组成。

[3] 事实上，这个字符串是“*we live in jupiter. we love jupiter.*”我们只是将字符做了相应的变换（包括空格和句号）：EMZBRCNKX 是“*jupiter.* ”（注意句号和空格），ACX 是“*we* ”，而 LPIC 是“*love*”，LGIC 是“*live*”。最后强调一点，光就原始的这串序列而言，我们是不可能知道 LPIC 和 LGIC 是否有明确语义，因为它们一次都没有重复过；而写成英文单词之后我们知道它的语义，是因为这两个单词其实在这句话和大脑构成的整体中是有重复的（如果大脑没见过这两个单词，我们也不能知道它们的语义，跟一串随机的字符在语义上并没有区别）。

该偏序多重集中的元素都称为“**组件**”。基础集中的元素也称为“**基础组件**”，最终需要生成的字符串 $\mathcal{X}$ 称为“**目标组件**”。注意，如上所示，基础集中基础组件的重数都是 0，且是无序的。然后，我们对这类偏序多重集定义一种“生成操作”:

■ “**生成操作**”(Generation-operation) 是指，取此类偏序多重集中的若干组件，其重数相应地减少（注意，可以取集合中的任何组件，即使其重数是零或负数），再以某种方式结合在一起（注意，新生成的组件必须是现集合中没有的），最后将结合而成的新组件放回到该集中，放在其构成组件最高层次的下一层。经过这个操作，就产生了一个新的偏序多重集。

注意，（1）该操作与 [29, 33] 中的结合过程不一样，因为该操作允许任意数量的组件结合，并且放回的时候保留了偏序。（2）这里的“负重数”只是暂时的记号，最终的结果中并不会出现负的重数。（3）这里“某种方式”对不同的系统可以有不同的定义。比如，对此例中的字符串系统，我们将其定义为将组件从左到右并排写在一起；对于分子，可以有一系列不同的生成操作，用以抓住分子的异构性等特征（见第 4.3节中更多的讨论）。（4）“放在其构成组件最高层次的下一层”是指，比如新组件由三个组件构成，而这三个组件分别处于第 1 层、第 3 层、第 5 层，那么须将新组件放在第 5 层后面，即第 6 层。

举例来更进一步说明。取 $\mathbf{S}_0$ 中的组件 A, C 结合成 AC 并放回，则称对 $\mathbf{S}_0$ 进行了一次生成操作，可记作 $\mathbf{S}_0$:A+C=AC $\to$ $\mathbf{S}'$={ A(-1), B(0), C(-1), D(0), E(0), F(0) // AC(1) }。注意，AC 放在了 A 和 C 的下一层，即，用“//”隔开。

再比如，取 $\mathbf{S}'$ 中的组件 D, D, F, AC 结合成 DDFAC 并放回，则称对 $\mathbf{S}'$ 进行了一次生成操作，可记作 $\mathbf{S}'$:D+D+F+AC =DDFAC $\to$ $\mathbf{S}''$={ A(-1), B(0), C(-1), D(-2), E(0), F(-1) // AC(0) // DDFAC(1) }。因为组件 D, F 和 AC 中，AC 处在更高的第 2 层，故 DDFAC 放在了第 3 层。

再比如，取 $\mathbf{S}''$ 中的组件 B, C 结合成 BC 并放回，则称对 $\mathbf{S}''$ 进行了一次生成操作，可记作 $\mathbf{S}''$:B+C=BC $\to$ $\mathbf{S}'''$={ A(-1), B(-1), C(-2), D(-2), E(0), F(-1) // AC(0), BC(1) // DDFAC(1) }。注意，BC 放在了第 2 层，因为 B 和 C 都是在第 1 层。

现在回到最初的获得 $\mathcal{X}$ 的问题上来。下面的例 1 中，通过连续几次生成操作，我们最终获得了目标组件 $\mathcal{X}$:

例 1. 第 1 次生成操作，$\mathbf{S}_0$:C+D=CD $\to$ $\mathbf{S}_1$={ A(0), B(0), C(-1), D(-1), E(0), F(0) // CD(1) }

. 第 2 次，$\mathbf{S}_1$:B+CD=BCD $\to$ $\mathbf{S}_2$={ A(0), B(-1), C(-1), D(-1), E(0), F(0) // CD(0) // BCD(1) }

. 第 3 次，$\mathbf{S}_2$:E+F=EF $\to$ $\mathbf{S}_3$={ A(0), B(-1), C(-1), D(-1), E(-1), F(-1) // CD(0), EF(1) // BCD(1) }

. 第 4 次，$\mathbf{S}_3$:A+BCD+BCD+BCD+CD+EF+EF=$\mathcal{X}$ $\to$ $\mathbf{S}_4$ = { A(-1), B(-1), C(-1), D(-1), E(-1), F(-1) // CD(-1), EF(-1) // BCD(-2) // $\mathcal{X}$(1) }

. 最后，从 $\mathbf{S}_4$ 中取出一个 $\mathcal{X}$，即达到我们的最终目的：获得了一个目标组件 $\mathcal{X}$。这最后一步可视为一次特殊的生成操作，即只取出而不放回，可记为
$\mathbf{S}_4$:$\mathcal{X}$(-1) $\to$ $\mathbf{S}_5$ = { A(-1), B(-1), C(-1), D(-1), E(-1), F(-1)//CD(-1), EF(-1) // BCD(-2) // $\mathcal{X}$(0) }

事实上，只要知道 $\mathbf{S}_0$ 和 $\mathbf{S}_1$，那么可以完全确定其生成操作是 C+D = CD。普遍地，知道 $\mathbf{S}_i$ 和 $\mathbf{S}_{i+1}$ 亦然。所以，我们达到目的的这条路径可简记为 $\mathbf{S}_0 \to \mathbf{S}_1 \to \mathbf{S}_2 \to \mathbf{S}_3 \to \mathbf{S}_4 \to \mathbf{S}_5 \Rightarrow \mathcal{X}$。显而易见地，有很多其他的路径也能同样达到目的，比如，

例 2. 第 1 次生成操作，$\mathbf{S}_0$:D+B+C=DBC $\to$ $\mathbf{S}'_1$={ A(0), B(-1), C(-1), D(-1), E(0), F(0) // DBC(1) }

. 第 2 次，$\mathbf{S}'_1$:E+F=EF $\to$ $\mathbf{S}'_2$={A(0), B(-1), C(-1), D(-1), E(-1), F(-1) // DBC(1), EF(1)}

. 第 3 次，$\mathbf{S}_2'$:A+B+C=ABC $\to$ $\mathbf{S}_3'$ = {A(-1), B(-2), C(-2), D(-1), E(-1), F(-1) // DBC(1), EF(1), ABC(1)}

. 第 4 次，$\mathbf{S}_3'$:DBC+DBC=DBCDBC $\to$ $\mathbf{S}_4'$={ A(-1), B(-2), C(-2), D(-1), E(-1), F(-1) // DBC(-1), EF(1), ABC(1) // DBCDBC(1) }

. 第 5 次，$\mathbf{S}_4'$:C+D=CD $\to$ $\mathbf{S}_5'$={ A(-1), B(-2), C(-3), D(-2), E(-1), F(-1) // DBC(-1), EF(1), ABC(1), CD(1) // DBCDBC(1) }

. 第 6 次，$\mathbf{S}_5'$:ABC+DBCDBC+D+CD+EF+EF=$\mathcal{X}$ $\to$ $\mathbf{S}_6'$={ A(-1), B(-2), C(-3), D(-3), E(-1), F(-1) // DBC(-1), EF(-1), ABC(0), CD(0) // DBCDBC(0) // $\mathcal{X}$(1) }

. 最后，$\mathbf{S}_6'$:$\mathcal{X}$(-1) $\to$ $\mathbf{S}_7'$={ A(-1), B(-2), C(-3), D(-3), E(-1), F(-1) // DBC(-1), EF(-1), ABC(0), CD(0) // DBCDBC(0) // $\mathcal{X}$(0) }

这条路径可记为 $\mathbf{S}_0 \to \mathbf{S}_1' \to \mathbf{S}_2' \to \mathbf{S}_3' \to \mathbf{S}_4' \to \mathbf{S}_5' \to \mathbf{S}_6' \to \mathbf{S}_7' \Rightarrow \mathcal{X}$。

2.3 定义“梯径”

■ 对于一条使我们达到最终目的的路径 $\mathbf{S}_0 \to \mathbf{S}_1 \to \mathbf{S}_2 \to \cdots \to \mathbf{S}_n \Rightarrow \mathcal{X}$，我们以如下的方法来构造另一个偏序多重集 J：取最终的集合 $\mathbf{S}_n$，删掉所有重数为零的元素，然后将所有的负重数取绝对值作为对应组件的重数，并保留所有的偏序不变。该过程构造出的这个偏序多重集 J，即定义为目标组件 $\mathcal{X}$ 的对应于这条路径的“**梯径**”(Ladderpath)。

■ 梯径中的组件都称为“**梯元**”(Ladderon)。

所以，例 1 所对应的梯径是：

$$J_{\mathcal{X},1} = \{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}, \mathrm{E}, \mathrm{F} \mathbin{/\!/} \mathrm{CD}, \mathrm{EF} \mathbin{/\!/} \mathrm{BCD}(2)\} \tag{1}$$

注意，梯径中的重数“(1)”通常省略不写。类似地，例 2 所对应的梯径是：

$$J_{\mathcal{X},2} = \{\mathrm{A}, \mathrm{B}(2), \mathrm{C}(3), \mathrm{D}(3), \mathrm{E}, \mathrm{F} \mathbin{/\!/} \mathrm{DBC}, \mathrm{EF}\} \tag{2}$$

显然地，每一条生成目标组件的路径都可以写成一条梯径，但反之则不尽然，比如，

例 3. 第 1 次生成操作，$\mathbf{S}_0$:D+B+C=DBC $\to$ $\mathbf{S}_1'$={ A(0), B(-1), C(-1), D(-1), E(0), F(0) // DBC(1) }

. 第 2 次，$\mathbf{S}_1'$:E+F=EF $\to$ $\mathbf{S}_2'$={A(0), B(-1), C(-1), D(-1), E(-1), F(-1) // DBC(1), EF(1) }

. 第 3 次，$\mathbf{S}_2'$:A+B+C+DBC+DBC+D+C+D+EF+EF=$\mathcal{X}$ $\to$ $\mathbf{S}_3''$={ A(-1), B(-2), C(-3), D(-3), E(-1), F(-1) // DBC(-1), EF(-1) // $\mathcal{X}$(1) }

. 最后，$\mathbf{S}_3''$:$\mathcal{X}$(-1) $\to$ $\mathbf{S}_4''$={ A(-1), B(-2), C(-3), D(-3), E(-1), F(-1) // DBC(-1), EF(-1) // $\mathcal{X}$(0) }

这条路径可记为 $\mathbf{S}_0 \to \mathbf{S}_1' \to \mathbf{S}_2' \to \mathbf{S}_3'' \to \mathbf{S}_4'' \Rightarrow \mathcal{X}$。可以看出，它也对应于梯径 $J_{\mathcal{X},2}$。为了更好地阐述梯径的定义，请参见附录 B 中的更多例子。

另外，值得指出

■ 任何组件都有至少一条“**平凡梯径**”，即完全由基础组件作为梯元构成的梯径。

比如，$\mathcal{X}$ 的平凡梯径是 $J_{\mathcal{X},0} = \{\mathrm{A}, \mathrm{B}(3), \mathrm{C}(4), \mathrm{D}(4), \mathrm{E}(2), \mathrm{F}(2)\}$，它代表了很多以任意顺序、由单个字符简单拼接而成的路径。

定义梯径的重要意义正是在于一条梯径能代表很多同等效的路径，它事实上完全过滤掉了生成目标组件过程中的冗余步骤信息。梯径中所保留下来的梯元及其重数、和梯元之间的偏序关系完整并且无冗余地刻画了所有等效的路径。梯径有几个很好的性质：

■ 梯元其实就是被重复利用过（即至少出现在 2 次生成操作中）的组件。正是由于这些重复利用，使得最后获得目标组件的过程能被简化（下文会看到，这种简化跟目标组件所包含的“信息”是紧密相关的）。

■ 对任一梯径 $J_{\mathcal{X}}$，

$$N_a = \sum_{i \in J_{\mathcal{X}}} (m_i \times n_{i,a}) \tag{3}$$

对每一个字符都成立。其中，N_a 是在目标组件 $\mathcal{X}$ 中字符 a 出现的次数，i 表示该梯径 $J_{\mathcal{X}}$ 中所有的梯元，m_i 是梯元 i 的重数，$n_{i,a}$ 是在梯元 i 中字符 a 出现的次数。

以 $J_{\mathcal{X},1}$ 为例，在目标组件 $\mathcal{X}$ 中，字符 B 出现了 3 次，则 $N_{\mathrm{B}} = 3$。在 (3) 式的右手边，梯元 BCD 的贡献是 $2 \times 1 = 2$（因为其重数是 2，且字符 B 在 BCD 中出现了 1 次）；梯元 B 的贡献是 $1 \times 1 = 1$（因为其重数是 1，且字符 B 在 B 中出现了 1 次）；其他梯元的贡献是 0。所以，等式右手边也等于 $2 + 1 = 3$。很容易验证式 (3) 对其他字符 A、C、D、E 和 F 都成立。类似地，我们可以验证梯径 $J_{\mathcal{X},2}$ 和 $J_{\mathcal{X},0}$。

任一梯径都能被完备地表示为一个偏序多重集（即偏序多重集表示，如式 (1) 和式 (2)）。但有时候将梯径用数学的图的形式表示可能更为直观，称为“**梯图**”，如下图 1a 所示：其中的层次即为梯径的偏序关系，组件之间的连接表示生成操作（比如 E 和 F 向上连接到 EF，表示 EF 由 E 和 F 生成）。

为了让梯图更简洁，我们作如下约定：

- 与基础组件相连的线都作淡化处理。
- 若下层组件 i 能够通过其他组件向上连接到上层组件 j，那么即使 i 与 j 直接相连，也无需画出 i 与 j 之间的连线。比如图 1a 中，下层组件 CD 应该直接连接到上层的目标组件 $\mathcal{X}$，因为梯径 $J_{\mathcal{X},1}$ 中，CD 会直接参与 $\mathcal{X}$ 的构造，但由于 CD 连接着 BCD，而 BCD 又连接着 $\mathcal{X}$，所以无需再画一条线连接 CD 和 $\mathcal{X}$。
- 各组件的重数可以标出（图 1中已标出），有时也可以不标出。

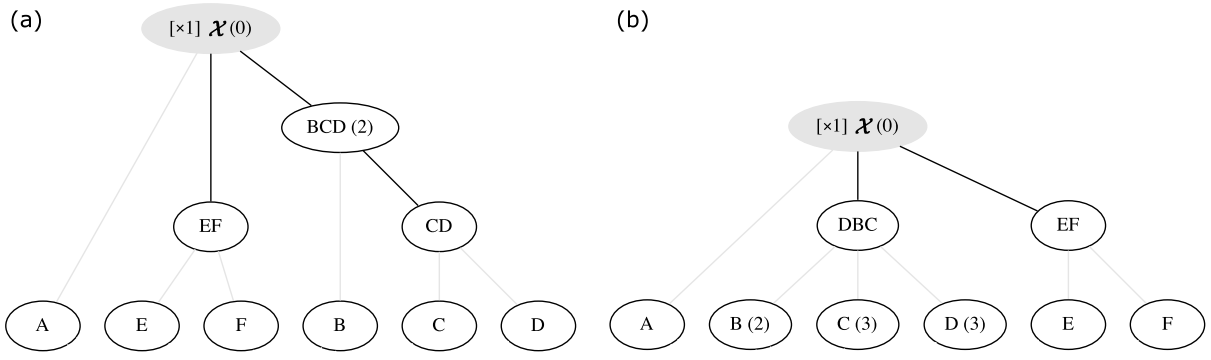


图 1:（a）梯径 $J_{\mathcal{X},1}$ 所对应的梯图。（b）梯径 $J_{\mathcal{X},2}$ 所对应的梯图。灰色用来表示是目标组件。其中，目标组件前面的“[×1]”表示“最后需要获得 1 个目标组件”。某组件后面如果加了 (n) 的话，则表示在该梯径（偏序多重集表示）中该组件对应的重数是 n，没有加 (n) 则表示其重数为 1。目标组件后面加的 (0) 表示在该梯径中其重数为 0（原则上不应该把重数为 0 的组件画出来，而这里画出来是为了方便读者理解目标组件和其他组件的层次关系）。

原则上，梯径的偏序多重集表示和梯图表示是一一对应的，但由于 (2) 和 (3) 的简化约定，可能使得梯图丢失掉一些信息。但不管怎样，梯径中最重要的信息在梯图中是得以保存的，即梯元之间的层次关系。梯径的偏序多重集表示是完备表述，略微侧重于表述目标组件的构成，而梯图表示更侧重于描述梯元之间的层次关系。

2.4 定义梯径度 λ、有序度 ω、规模度 S

这一节我们将引入非常重要的概念：梯径度 (ladderpath-index)，和与之相关的概念，规模度 (size-index) 和有序度 (order-index)。但在此之前，我们需要先定义“梯径的单位长度”和“梯径的长度”。

- 对于上文中的字符串的例子，我们将“**梯径的单位长度**”定义为：将任意 2 个字符串（即组件）并排写在一起的一次操作。一个单位长度称为 1“**程** (*lift*)”。（注意的是，对于不同的系统，如何结合组件也可能是不同的，所以梯径单位长度的定义也可以是不同的，不过我们都将一个单位长度称为“程”）。

由于一次生成操作实际上是将 n 个组件并排写在一起，也就相当于“将 2 个组件并排写在一起”这个操作实施了 $(n-1)$ 次，所以，每次生成操作实际上就对应了 $(n-1)$ 程。比如，在上面例 1 的路径 $\mathbf{S}_0 \to \mathbf{S}_1 \to \mathbf{S}_2 \to \mathbf{S}_3 \to \mathbf{S}_4 \to \mathbf{S}_5 \Rightarrow \mathcal{X}$ 中，第 1 次生成操作 C+D = CD 的长度是 1 程，第 2 次 B+CD = BCD 的长度是 1 程，第 3 次 E+F = EF 的长度是 1 程，第 4 次 A+BCD+BCD+BCD+CD+EF+EF = $\mathcal{X}$ 的长度是 6 程。对于最后一次特殊的生成操作，我们可以将其看成是将 $\mathcal{X}$ 和一个假想的空字符并排写在了一起，故其长度是 1 程。在例 2 的路径中，每一次生成操作的长度分别是 2、1、2、1、1、5、1 程。在例 3 的路径中，每一次生成操作的长度分别是 2、1、9、1 程。

注意，前面已经提到，梯径的单位长度可以以不同的方式定义。比如，在研究化学分子时，可以将原子、离子、分子片段之间生成一个化学键定义为单位长度；在研究进化时，可将一个物种的出现、或是一种功能的出现定义为单位长度；等等。所以，梯径的单位长度是一个由用户定义（user-defined）的量，但是一旦定义好，在后面的分析中就不能再改变了。

那么现在，我们可以作进一步定义：

- “**梯径（J）的长度**”定义为该梯径中所有生成操作的长度之和，记为 $|J|$。注意，任一条路径的长度也就自然地定义为其所有生成操作的长度之和。

$|J|$ 是用来衡量对于这条梯径，生成目标组件需要“花费多少代价”。一条梯径往往对应很多条生成目标组件的路径，但所有这些路径的长度都是相同的（这正是梯径的一个良好性质），故选取任意一条路径来计算即可。比如，根据定义我们可以计算出例 2 和例 3 的路径长度都是 13 程，而它们也确实都对应于同一条梯径 $J_{\mathcal{X},2}$（当然，也可以得出 $|J_{\mathcal{X},2}| = 13$ *lifts*）。对于例 1 的路径来说，其长度是 10 程，故其对应的梯径 $J_{\mathcal{X},1}$ 的长度是 10 程（比 $J_{\mathcal{X},2}$ 短）。

现在，我们可以给出“梯径度”的定义了：

- 目标组件 $\mathcal{X}$ 的“**梯径度**”定义为 $\mathcal{X}$ 的**最短梯径**（最短梯径可能有一条，也可能有几条）的长度，记为 $\lambda(\mathcal{X})$。显然，其单位也为“程 (*lift*)”。

注意：（1）梯径度跟文献 [33, 31] 中的“assembly index”是不同的，因为后者测量的是组合操作的步骤数，而前者测量的是生成操作的长度，且对于不同的系统，对生成操作的单位长度的定义也不相同。（2）计算最短梯径并不简单。我们已经证明它至少跟一个 NP 完全问题一样困难（即，随着问题尺度的增加，该问题不可能在多项式时间内被解决），不过我们仍然发展出了一套严格的流程和算法，详见第 2.7节。

对目标组件 $\mathcal{X}$ 来说，$J_{\mathcal{X},1}$ 其实正是它的最短梯径，所以 $\mathcal{X}$ 的梯径度就是 10 程，即 $\lambda(\mathcal{X}) = 10$ *lifts*。事实上，对任一目标组件 $\mathcal{X}$ 来说，梯径度 $\lambda(\mathcal{X})$ 描述了 $\mathcal{X}$ 的一个本征特性，即，生成 $\mathcal{X}$ 所需的最小“代价”。这些最短梯径往往是我们最关心的，下文会详细讲到，它们对应着 $\mathcal{X}$ 最紧凑地储存信息的方式。

现在，我们给出第二个重要概念，即“规模度”：

■ 一个组件 i（比如目标组件、梯元）的“**规模度**”定义为其最短的平凡梯径（可能是一条，也可能是几条，但它们都有相同的长度）的长度，记为 $S(i)$。

对于字符串来说，如果基础组件是单个字符，那么其规模度就是字符串中字符的个数，比如 $\mathcal{X}$ 的规模度是 16 程，因为由单个字符拼接而成即对应着它的平凡梯径。

为什么定义中要强调最短的平凡梯径呢? 因为如果基础集中有多个字符组成的字符串，比如，若 $\mathcal{X}$ 的基础集是 { A, B, C, D, E, F, BCD }，那就可能有不同的平凡梯径。可以是像上面那样只用单个字符来拼接，即梯径 { A, B(3), C(4), D(4), E(2), F(2) }，其长度是 16 程；也可以是利用基础集中的字符串，即梯径 { A, BCD(3), C, D, E(2), F(2) }，其长度是 10 程。这个时候我们就必须在所有这些平凡梯径中选择最短的长度作为其规模度。

然后，我们给出“有序度”的概念：

■ 目标组件 $\mathcal{X}$ 的“**有序度**”定义为

$$\omega(\mathcal{X}) := S(\mathcal{X}) - \lambda(\mathcal{X}) \tag{4}$$

其中 $S(\mathcal{X})$ 是 $\mathcal{X}$ 的规模度，$\lambda(\mathcal{X})$ 是 $\mathcal{X}$ 的梯径度。显然，有序度的单位也是“程 (*lift*)”。

也就是说，有序度等价于通过最短梯径来生成该目标组件所节约的“长度”，即所节约出来的结合组件的操作所花费的那些步骤、代价。这是符合直觉的：因为节约的步骤、代价（即“程”数）越多，也就表明目标组件越有序。特别说明，这里的“有序的”实际上等同于“有组织的”，取英文中“organized”的意思。

最后，我们给出梯径长度的一个重要性质，这对于计算梯径度、有序度极为重要：

■ 任一梯径 $J_{\mathcal{X}}$ 的长度可以直接由它的偏序多重集表示算出，而无需将其转化成某一条对应的路径再计算。计算方法是：

$$|J_{\mathcal{X}}| = S(\mathcal{X}) - \sum_{i \in J_{\mathcal{X}}} m_i \cdot (S(i) - 1) \tag{5}$$

其中，$S(\mathcal{X})$ 是该目标组件 $\mathcal{X}$ 的规模度，$S(i)$ 是梯元 i 的规模度，m_i 是梯元 i 的重数，i 代表该梯径 $J_{\mathcal{X}}$ 中所有的梯元。

比如，在例 1 的梯径 $J_{\mathcal{X},1}$ = { A, B, C, D, E, F ⫽ CD, EF ⫽ BCD(2) } 中：$\mathcal{X}$ 的规模度是 16 程；基础组件的规模度都是 1 程（所以基础组件的贡献总是 0，总不用考虑）；CD、EF、BCD 的规模度分别是 2、2、3 程；所以 $|J_{\mathcal{X},1}| = 16-(2-1)-(2-1)-(3-1)\times 2 = 10$ *lifts*。同样地，$|J_{\mathcal{X},2}| = 16-(3-1)-(2-1) = 13$ *lifts*。可以验证，它们都跟用定义算出的结果是一样的。

■ 事实上，根据(5)式可以推导出，目标组件 $\mathcal{X}$ 的有序度可以方便地由下式算出：

$$\omega(\mathcal{X}) = \sum_{i \in J_{\mathcal{X}}} m_i \cdot (S(i) - 1) \tag{6}$$

其中 i 代表最短梯径中所有的梯元。

2.5 梯径度和有序度是“复杂性”的两个轴

任何一个系统 $\mathcal{X}$ 的最短梯径 $J_{\mathcal{X}}$ 本身包含了描述该系统 $\mathcal{X}$ 的复杂性、及其所蕴含的信息量的“全部知识”。而由最短梯径所计算出的梯径度和有序度则是这种全部知识在某些方面的概括：

■ 系统 $\mathcal{X}$ 的梯径度 $\lambda(\mathcal{X})$ 描述了 $\mathcal{X}$ 所蕴含的“信息”的多少，即为了生成 $\mathcal{X}$ 需要“外界”额外花费多少代价、放入多少“信息”；也就相当于生成 $\mathcal{X}$ 有多困难。这里的“信息”跟“信息熵”、“热力学熵”所指的“信息”不太一样。

- 系统 $\mathcal{X}$ 的有序度 $\omega(\mathcal{X})$ 实际上描述了有多少"信息"是可以被节约出来的（即平凡梯径跟最短梯径之间的差值），即有多少冗余的"信息"。这是符合直觉的，因为能节约的步骤、代价（即"程"数）越多，也就表明 $\mathcal{X}$ 越有序。

- 现在，我们能看出，我们直觉上所说的"复杂"正是包含了这两方面，一方面由梯径度 λ 刻画，其侧重于描述构造该系统的代价和困难；另一方面由有序度 ω 刻画，其侧重于描述该系统是怎样有序地、以层级（hierarchy）的方式构造起来的。

- 最后，对于一个特定的目标组件 $\mathcal{X}$（或目标系统），其梯径度和有序度之和恒等于其规模度，即 $\lambda(\mathcal{X}) + \omega(\mathcal{X}) \equiv S(\mathcal{X})$，这自然而然地解决了不同规模、大小的系统之间"归一化"的问题。

为了阐述得更清楚，我们现考虑一个简单例子：考虑第 2.2节中的 $\mathcal{X} =$ ABCD BCDBCDCDEFEF，其有一定的结构，和另一串相同长度的随机字符 $\mathcal{W} =$ ABCDEFCFEDCBADBA。直觉上来说，一方面，随机的 $\mathcal{W}$ 所包含的信息量比不随机的 $\mathcal{X}$ 多，因为，若要复述 $\mathcal{W}$，需要的信息量（即需要记忆的信息）比 $\mathcal{X}$ 大；另一方面，随机的 $\mathcal{W}$ 比不随机的 $\mathcal{X}$ 无序得多。这两个方面与用以上 λ 和 ω 的概念来描述是吻合的：一方面，$\lambda(\mathcal{W}) = 16 > \lambda(\mathcal{X}) = 10$，即表示 $\mathcal{W}$ 蕴含的"信息"比 $\mathcal{X}$ 多；另一方面，$\omega(\mathcal{W}) = 0 < \omega(\mathcal{X}) = 6$，即表示 $\mathcal{W}$ 比 $\mathcal{X}$ 无序。

上面的这些概念不仅仅适用于字符串，对任何实体都适用。根据定义有序度的(4)式，我们可以作图 2a，其中，每一根斜线就是规模度 S 的等高线。可以看出，同样梯径度（即蕴含同样多信息，有同样的难度）的情况下，一个系统越有序，则它的规模越大；而同样规模的情况下，一个系统越有序，则它的梯径度越小，即蕴含的信息越少，构造它的难度越小。

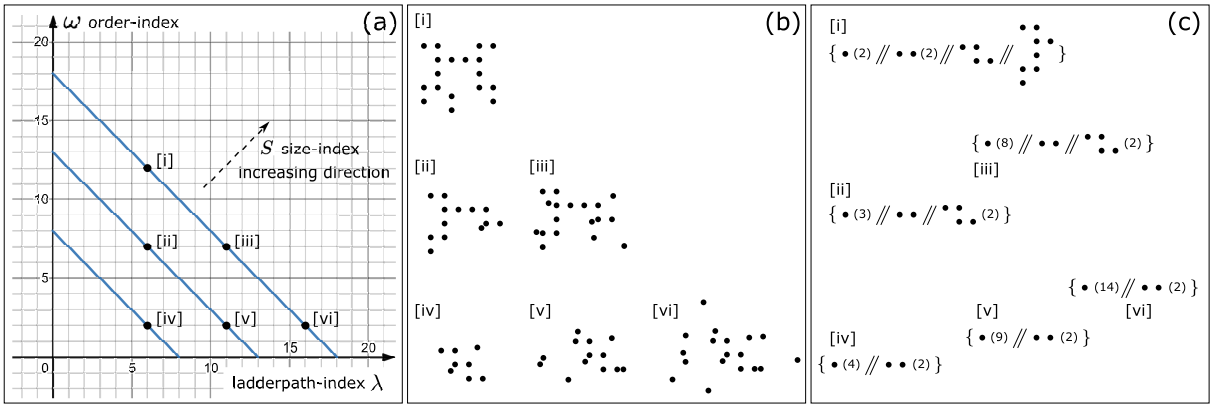


图 2: （a）梯径度 λ、有序度 ω、规模度 S 之间的关系。蓝色斜线是 S 的等高线。[i]-[vi] 对应着图 b 中的图案（注意，一个坐标可以对应无穷种图案），其坐标分别为 $(6,12), (6,7), (11,7), (6,2), (11,2), (16,2)$。（b）对应于图 a 中的 6 个坐标上的图案。（c）该 6 个图案的梯径。

现在，再通过一个详细的例子来进一步阐述梯径的概念是如何描述客体的"复杂性"的。考虑图 2b 中的 6 个图案，其可以想象成一块空地上的石头的排布，即这些小黑点。这些图案能够以如图 2c 所示的梯径描述（附录 C 中详细阐述了如何计算这些图案的梯径），其各自的梯径度和有序度亦可以由此计算。

从图 2中可以看出以下几点[4]：

1. 在同样规模的情况下（比如 [i], [iii] 和 [vi]），随着 ω 的增加，图案变得越来越有序，这是很符合直觉的；随着梯径度 λ 的增加，图案变得越来越难以复现（比如若要复现 [vi]，那么需要记

[4]值得提额外的一点，有序度 ω 有一个很好的性质：我们可以给目标组件添加假想的完全无序成分，使其规模度变大，但这个操作却不会使其原本的 ω 发生变化。比如在图 2b 中，由于 [ii] 和 [iii] 的规模不一样，所以似乎不知道该怎么比较，但我们可以假想给 [ii] 完全无规律地放上额外 5 块石头，那么此时 [ii] 和 [iii] 就有同样的规模了，且 [ii] 的有序度 ω 还是 7 程（因为其梯径度并没有变）。也就是说，为了符合直觉，我们可以事先把所有需要比较的目标都假想成同样的规模再进行比较；不过方便的是，由于 ω 的这个良好的性质，它所描述的有序度其实已经把目标规模的影响扣除了，所以要不要事先转换成在直觉上同样的规模都无所谓。

住几乎每一块石头的位置，而复现 [i] 则不需要记住每块石头的位置，只需要记住梯径的描述），这也符合直觉。

2. 我们也可以看出一些反直觉的地方，但这也正是最关键的地方：比如，[i] 比 [ii] 有序，但生成两者的难度其实是一样的，因为它们的 λ 相同；[i] 比 [iii] 和 [vi] 有序，但生成 [i] 的难度却更小，需要的程数更少；[i] 比 [v] 有序，但不仅生成 [i] 的难度更小，而且 [i] 的规模还比 [v] 大（用 [ii] 跟 [iv], [v] 和 [vi] 比也是同样的道理，以此类推）。下文可以看到，这最后一点也是我们解释生命这个有序系统的起源并不是我们想象中那么困难的关键。

值得一提的是，直觉上我们可能有时会说 [vi] 比 [i]“复杂”，因为前者看上去更混乱、不规则、复现起来更困难；可能有时又会说 [i] 比 [vi]“复杂”，因为 [i] 需要更详细、复杂的、微妙的机制来生成[5]。但不难看出，这两个“复杂”其实是指向了不同的方面。前者对“复杂”的解读其实是关于系统蕴含了多少信息（等价地，复现该系统有多困难、或需要额外加入多少信息）；后者的解读其实是关于系统有多有序。事实上，这两方面正好对应了梯径度 λ 和有序度 ω，所以由此我们能够区分出“复杂”的这两个轴。

最后需要说明的是，定义式 $\lambda(\mathcal{X})+\omega(\mathcal{X})\equiv S(\mathcal{X})$ 表明对于一个特定的目标组件或目标系统 $\mathcal{X}$ 而言，复杂性的两个轴不是独立的。确实如此，但是任一目标都能被放在这套坐标的任一位置上，因为尽管这三个指标被 $\lambda+\omega\equiv S$ 所约束，但其中任意两个都是自由的。参见图 2，如果我们固定目标组件的规模，通过重排其图案、排布，该目标组件会在该套坐标中自由地移动（想像在图案 [i]、[iii] 和 [vi] 中重排）。正是因为目标组件内部的结构和信息，才使得其被固定在坐标中的确定位置。其实我们也可以选择 λ 和 S，或者 ω 和 S 作为轴，但我们这里的动机是将对复杂性的直觉与坐标轴联系起来，而正如上面所讨论的，我们对于复杂性的直觉往往来自于困难程度和秩序。值得强调的是，规模是一个客体复杂度的重要指标，但并不是简单的正比关系——并不是规模越大则越复杂——比如，参见图 2b，图案 [i] 比 [v] 大但是 [i] 更容易被复现（因为 [i] 的梯径度更小）。正因为如此，我们需要 λ 和 ω 来表示复杂度的两个轴。有了这些概念，我们现在就能比较容易地比较不同规模的客体的复杂度了。

2.6 概念推广：整个系统的梯径

截止目前，我们讨论的都是单个目标组件的梯径，但实际上所有这些概念都可以直接推广到整个系统。为此，事实上，我们只需将所有需要处理的组件看作一个大的“目标组件”，为方便起见，称其为“**目标系统**”。比如，这个系统里有 2 个字符串 AAB，3 个字符串 BC，和 1 个字符串 DDF，那就可以把它们看成一个大的“目标组件”，即字符串“AAB, AAB, BC, BC, BC, DDF”，该粘起来的字符串则为“目标系统”（注意，这些单个字符串之间并没有连在一起，跟真正的一个单一的字符串 AABAAB-BCBCBCDDF 是有区别的）。顺带一提，目标系统梯径的概念跟文献 [33] 中的“molecular assembly tree”是不一样的，后者只能处理一组完全不同的分子，而前者并没有此限制。

现举一个例子来说明目标系统的梯径（其他概念的推广也就显而易见了）。假设我们需要获得的目标系统是

$$\mathcal{Q}=\{\mathrm{ABDEDBED}(2),\mathrm{ABDED},\mathrm{ABDABD},\mathrm{CAB}(2),\mathrm{ED}(3)\}$$

其中，其基础集是 $\mathbf{U}_0$ = {A(0), B(0), C(0), D(0), E(0)}。那么下面这一系列生成操作就代表了一条路径：

[5]这其实落入了一个直觉上的陷阱：我们觉得生成 [vi] 的机制很简单是因为我们可以直接随便用手一甩就能甩出一种随机的石头的排布。然而如果要真正复现出 [vi] 那样的排布，需要知道每一块石头的位置，原则上需要 18 程，但 [vi] 中确实也有重复的结构，所以根据梯径，最少需要 16 程；而为了复现出 [i] 的排布，只需要 6 程（按照最短梯径所给出的结果）。所以，实际上 [i] 所蕴含的信息是更少的。虽然“随手甩”这种机制操作起来很简单，但是对于最后生成的排布，实际上有很多随机过程都参与了其中（比如空气的流动、出力的大小等等），所以最后的排布拥有的信息量是很大的，要复现这种排布是不容易的，也就是说，生成 [vi] 的机制本质上是复杂的而不是像原本直觉上认为的简单的。

例 4. 第 1 次生成操作，$\mathbf{U}_0$:A+B=AB $\rightarrow$ $\mathbf{U}_1$={ A(-1), B(-1), C(0), D(0), E(0) // AB(1) }

第 2 次，$\mathbf{U}_1$:C+AB=CAB $\rightarrow$ $\mathbf{U}_2$={A(-1), B(-1), C(-1), D(0), E(0) // AB(0) // CAB(1) }

第 3 次，$\mathbf{U}_2$:AB+D=ABD $\rightarrow$ $\mathbf{U}_3$={ A(-1), B(-1), C(-1), D(-1), E(0) // AB(-1) // CAB(1), ABD(1) }

第 4 次，$\mathbf{U}_3$:ABD+ABD=ABDABD $\rightarrow$ $\mathbf{U}_4$={ A(-1), B(-1), C(-1), D(-1), E(0) // AB(-1) // CAB(1), ABD(-1) // ABDABD(1) }

第 5 次，$\mathbf{U}_4$:E+D=ED $\rightarrow$ $\mathbf{U}_5$={ A(-1), B(-1), C(-1), D(-2), E(-1) // AB(-1), ED(1) // CAB(1), ABD(-1) // ABDABD(1) }

第 6 次，$\mathbf{U}_5$:ABD+ED=ABDED$\rightarrow$ $\mathbf{U}_6$={ A(-1), B(-1), C(-1), D(-2), E(-1) // AB(-1), ED(0) // CAB(1), ABD(-2) // ABDABD(1), ABDED(1) }

第 7 次，$\mathbf{U}_6$:ABDED+B+ED=ABDEDBED $\rightarrow$ $\mathbf{U}_7$={ A(-1), B(-2), C(-1), D(-2), E(-1) // AB(-1), ED(-1) // CAB(1), ABD(-2) // ABDABD(1), ABDED(0) // ABDEDBED(1) }

最后，从 $\mathbf{U}_7$ 中取出目标系统 $\mathcal{Q}$ 中所包含的组件，即达到最终目的。这最后一步可视为一次特殊的生成操作，即只取出而不放回，可记为 $\mathbf{U}_7$:$\mathcal{Q}$(-1) $\rightarrow$ $\mathbf{U}_8$ = { A(-1), B(-2), C(-1), D(-2), E(-1) // AB(-1), ED(-4) // CAB(-1), ABD(-2) // ABDABD(0), ABDED(-1) // ABDEDBED(-1)}

所以，这条路径所对应的梯径是（其梯图表示见图 3）：

$$J_{\mathcal{Q}} = \{\mathrm{A}, \mathrm{B}(2), \mathrm{C}, \mathrm{D}(2), \mathrm{E} \,/\!/\, \mathrm{AB}, \mathrm{ED}(4) \,/\!/\, \mathrm{CAB}, \mathrm{ABD}(2) \,/\!/\, \mathrm{ABDED} \,/\!/\, \mathrm{ABDEDBED}\} \tag{7}$$

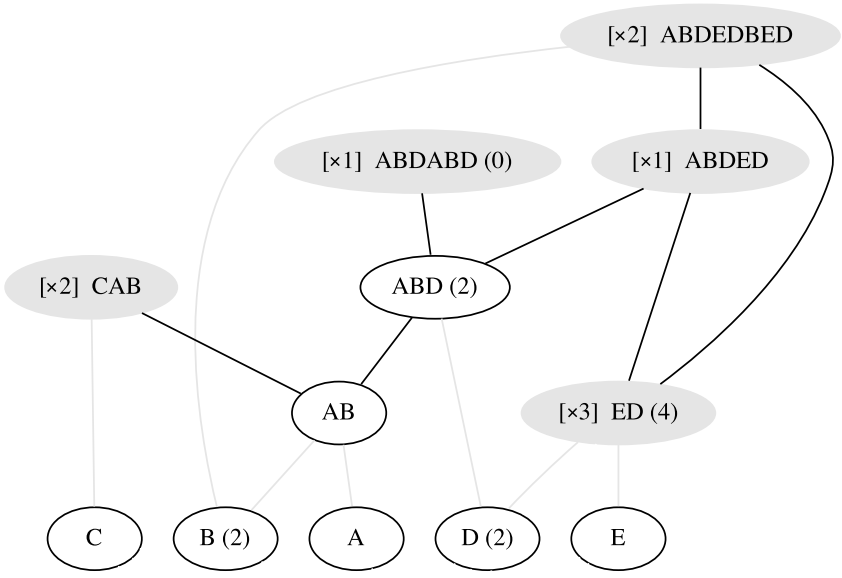


图 3: 目标系统 $\mathcal{Q}$ 的一条梯径 $J_{\mathcal{Q}}$ 的梯图表示（实际上是 $\mathcal{Q}$ 的最短梯径）。所有灰色的组件共同组成了这个目标系统 $\mathcal{Q}$，灰色组件前面的"[×n]"表示目标系统 $\mathcal{Q}$ 中包括 n 个该组件。组件后面如果加了 (n) 的话则表示在该梯径（偏序多重集表示）中该组件对应的重数是 n，没有加 (n) 则表示其重数为 1，如果是 (0) 则表示在该梯径中其重数为 0（原则上不应该把重数为 0 的组件画出来，而这里画出来是为了方便读者理解某些重要的组件间的层次关系）。

事实上，这条梯径正是 $\mathcal{Q}$ 的最短梯径（可由我们的算法算出，详见第 2.7节）。所以，根据(6)式，我们可以算出 $\mathcal{Q}$ 的有序度是 $\omega(\mathcal{Q}) = 1 + 1 \times 4 + 2 + 2 \times 2 + 4 + 7 = 22$ *lifts*。且 $\mathcal{Q}$ 的规模度是 $S(\mathcal{Q}) = 8 \times 2 + 5 + 6 + 3 \times 2 + 2 \times 3 = 39$ *lifts*（即 $\mathcal{Q}$ 中所有字符的总个数），所以也不难根据(4)式算出其梯径度是 $\lambda(\mathcal{Q}) = 39 - 22 = 17$ *lifts*。我们也可以根据该梯径所对应的路径中所有生成操作的

长度之和来验证：前 6 次生成操作的长度都是 1 程，第 7 次生成操作的长度是 2 程，最后一次特殊的生成操作的长度是 9 程，所以答案也是 17 程。

2.7 最短梯径的算法

为了最终找到最短的梯径，我们首先给出一个具体流程用以计算一条梯径，不论其长度。这里我们以目标系统而不以单个的目标组件为示范来阐释，因为前者是更为普遍的情形（不失一般性地，以字符串为例）：

1. 首先创建一个空的多重集 $\mathcal{H}$ 用以存放组件。在最后我们可以通过 $\mathcal{H}$ 轻松地计算出梯径。
2. 从目标系统 $\mathcal{Q}$ 出发，每个不同的组件都只保留一个在 $\mathcal{Q}$ 中，其余的重复组件全部放入 $\mathcal{H}$ 中。
3. 以某种预先规定的特定方式，持续切割（即，将字符串分割成任意多段子字符串或字符）$\mathcal{Q}$ 中的组件，直到 $\mathcal{Q}$ 中有至少 2 个子字符串或字符相同。然后在 $\mathcal{Q}$ 中只保留一个该字符串，其余重复的字符串全部放入 $\mathcal{H}$ 中。注意，“预先规定的特定方式”往往有很多种，参见附录 D 的其中一个例子。
4. 重复步骤 3，直到 $\mathcal{Q}$ 中不再能找到重复的字符串或字符。最后把剩下的全部切割成基础组件，然后全部放入 $\mathcal{H}$ 中。此时的 $\mathcal{H}$ 即记录了一条梯径。
5. 最后，需要将 $\mathcal{H}$ 中各字符串的层次关系（即，根据它们是如何被分割的：谁是父字符串，谁是子字符串）排列清楚，即最终得到该梯径的偏序多重集表示。

参见附录 D 的一个详细例子。上述流程的结果取决于如何切割字符串。如果遍历所有切割方式，我们就可以得到该目标系统的所有可能梯径，从而找到最短的。这也就是一个计算最短梯径的算法。该算法对应的代码可以在此下载https://github.com/yuernestliu/ladderpath

这里我们的动机是提供一个计算较小目标组件或目标系统的最短梯径的原型算法和代码，而不是提供一个高效和具有实用价值的算法以处理长序列（比如用于字符串压缩或基因序列分析）。我们的想法是，如果处理短字符串的算法可以被开发出来，那么其他处理长字符串、图像、分子、蛋白质、三维物体等的算法也能在未来被发展和臻善。另外值得一提的是，Lempel-Ziv 算法也是一个无损压缩算法 [12, 13]，所以它与我们这里计算梯径的算法在关于层次结构的大体概念和一些编程技巧上也有一些相似之处。但是尽管如此，除了动机不同以外，这两个算法给出的结果也是不一样的。

最后关于目前版本算法的一点评论：对于一个大型系统而言，遍历所有的梯径几乎是不可行的，因为切割字符串的方式随着其长度的增加呈指数增加。事实上，找出最短梯径至少和一个 NP 完全问题一样困难，因为它可以看作是 *addition chain* 问题（Knuth 在他的书中介绍过 [28]）的一个更复杂的版本，而该问题已被证明是一个 NP 完全问题 [34]。

尽管如此，在实践中，找出“足够短”的梯径可能是可以接受的，故而可能有捷径。比如，我们可以稍微将上述算法做一点修改：

- 上述步骤 3 中，我们优先寻找最长的重复子字符串。

这样一来，以这种方式得到的梯径虽然不能保证是严格最短的，但是它往往是足够短的。该直觉在于，梯元（即，重复的子字符串）越长，越多的生成操作可以被节约下来，所以梯径也就越短。

3 梯径在进化中的重要性：梯径系统

系统的梯径度 λ 对应了生成该系统所需花费的最小代价。如果假定任一生成操作所对应的概率都是一样的，那么 λ 就反比于生成该系统的总概率，即，λ 越大，最后生成该系统的概率就越小。比如，在图 2的例子中，生成/复现 [i] 要比生成/复现 [vi] 容易。

不过，“生成 [i] 比生成 [vi] 容易”这个结论要成立，我们必须有一个关键的假设。这个关键假设是：任一组件只要在前面的步骤中生成过，就能被无限量地重复利用，而无须从头生成（即，任一组件只要被生成，其个数立刻是无穷的，就像其有一个取之不尽的储备库，或等价地说，梯元的个数永远是无穷的）。比如，在 [i] 中，最后一个梯元（参见图 2c 中由 8 个黑点组成的图案）被重用了 1 次（即重复了 2 次），也就意味着节省了一个该图案的平凡梯径的长度（即 7 程）。正是因为这些重用，使得生成 [i] 比 [vi] 简单。

在某些真实情形下（而非所有情形），这种重用是可能的。比如考虑科技的时候，发明过的东西如果已经在社会上广为人知，那么就可以在以后的社会中重用而无须再发明，这时梯元的重用就是存在的；但如果某人发明了一个东西而其他人不知道，那么从全社会的角度来看，这个发明的重用就不可能，也就是说，该情形下这个系统（即整个社会）就不能用梯径来描述。

所以自然地，一个更进一步的问题是：能用梯径来描述的系统必须满足什么条件？答案是，需要两个条件：一是能够生成新组件，二是组件能够复制（进而数量可以增长）。第一个条件隐含在生成操作中，第二个条件隐含在“梯元取之不尽”的假设中。因此，

- 我们将一个系统称为“**梯径系统**”如果它满足这两个条件：（i）能够生成新组件，（ii）部分或所有组件能够复制。

有趣的是，这两个条件在很多情形和系统里都是成立的，比如生命、语言、和科技。

值得注意的是，对于第一个条件“能够生成新组件”，新组件并不一定要通过结合已有的两个组件的方式生成，即，生成操作不一定要像第 2.2节描述的那样，而是可以根据特定的系统而被不同地定义。比如，对于生命，如果考虑单个细胞，不仅基因重组和基因横向传播可以被定义为生成操作（这两种方式仍然属于结合已有组件），基因变异也可以被定义为生成操作（这就不同了，属于改变现有组件）。对于语言，人们通过组合旧词、借用别的语言等方式创造新的词语，从而产生新组件。对于科技，发明是产生新组件的方式，而发明往往是通过结合或修改已有的东西。

对于第二个条件“复制”，其可以指自我复制，也可以指在其他东西的帮助下复制。也就是说，组件要能成为梯元，必须要有复制的能力。所有梯元都是组件，但只有能够复制的组件才是梯元。哪些组件能够复制哪些不能为“自然选择”提供了产生重大作用的舞台。生命的一个显著特征就是能够自我复制：个体、物种可以自我复制，DNA 也可以复制（其实是 DNA、RNA 和蛋白质构成的网络能够复制 [35, 36, 37]）。对于语言，刚创造的新单词、新短语等都可以被后来的人再次利用，这也是复制的过程（在以前远距离的交流不是很方便的时候，被一个地方的人们发明的新单词可能不太容易在整个社会上传播，所以在全社会来看，这种新单词就没有能够成为梯元，这可能也正是语言分裂成各种方言、或演化成其他语言的机制）。对于科技，如果发明通过论文、专利等形式被详细记录着，使得其可以被再现，那么它就可以看成是能够复制的。

鉴于前面介绍了梯径系统的概念，我们应该可以讨论以下这个问题了：完全随机的结构（比如白噪声）携带了最多的信息，还是最少的信息？这里有两个不同的角度来考察这里所说的“信息”。（1）若要重现白噪声的一个特定序列，我们需要记住这整个序列，从这个意义讲，确实需要最大的信息量。（2）但是，复杂性并不仅仅可以指“重现有多困难”，也可以指更底层的跟重复和（自然）选择有关的层级结构信息，这是另一层面上的信息。我们主张，这第二层次的信息是由上述梯径系统的两个特性所导致的。即，这种跟（自然）选择有关的信息量大意味着梯径度和有序度同时很高。所以，这种意义下，白噪声几乎不含有这种信息，因为虽然其梯径度很高但是其有序度却非常低。

以此结束此节：生命得以起源，很可能是通过这种“梯径机制”（而不是某种“奇迹 magical event”），即，在起源、发展的过程中，能够复制的梯元一定被生成了很多很多。事实上，确实有这种证据：比如，细胞、物种显然可以自我复制；构成单个细胞或化学反应网络（比如自催化集合 autocatalytic set [35, 36, 37]）的分子很多也是能够复制的。

4 讨论

本文已经介绍了梯径的概念，故可以用梯径来刻画一个客体复杂度的两个方面：梯径度和有序度。现在我们讨论这个概念在两种不同情形下的应用："**孤立体系**"和"**联合体系**"。

4.1 关于信息：外星信号（孤立体系）

假设我们收到一串外星球发来的字符 $\mathcal{Y}$（或者是某种信号，转换成了这种字符串的形式）：

$$\mathcal{Y} = \text{TBCDEFRBCDEFTEFHKREFHJKLMUVTEFPSMU}$$

然后需要判断这里面是否包含了某种信息？首先，$\mathcal{Y}$ 可以被看作一个"**孤立体系**"，因为没有其他的字符串或者信号能够与 $\mathcal{Y}$ 一起被考虑。

利用梯径理论和我们已经开发出的算法（第 2.7节），$\mathcal{Y}$ 的最短梯径是

$$J_{\mathcal{Y}} = \{\text{T}, \text{B}, \text{C}, \text{D}, \text{E}, \text{F}, \text{R}(2), \text{H}(2), \text{K}(2), \text{J}, \text{L}, \text{V}, \text{T}, \text{P}, \text{S}, \text{M}, \text{U} \mathbin{/\!/} \text{EF}(2), \text{MU} \mathbin{/\!/} \text{TEF}, \text{BCDEF}\}$$

故其梯径度 $\lambda(\mathcal{Y}) = 34-1\times2-1-2-4 = 25$ 程。所以我们可以将 $\mathcal{Y}$ 理解成 T [BCD[EF]] R [BCD[EF]] [T[EF]] H K R [EF] H J K L [MU] V [T[EF]] P S [MU]，其中方括号将梯元间隔开。

虽然现在下结论说 $\mathcal{Y}$ 不是随机产生的还为时过早[6]，但至少可以看出，整个字符串是组织得很好很有序的（我们似乎能推测出 EF 一定是这句话里最关键的，因为它出现的次数最多；而 MU、TEF、BCDEF 也很重要）。事实上 $\mathcal{Y}$ 是这句话（忽略标点和大小写）：*my mother made a chocolate cake your mother made a chocolate cake my chocolate cake was delicious your chocolate cake was not delicious I ate half of my chocolate cake and you ate half*（将这句话记作 $\mathcal{Z}$）。$\mathcal{Y}$ 中的每个字符对应着 $\mathcal{Z}$ 里的一个单词。所以，EF 是 *chocolate cake*, TEF 是 *my chocolate cake*, BCDEF 是 *mother made a chocolate cake*, MU 是 *ate half* 。

有下面四点需要进一步说明。第一，$\mathcal{Y}$ 最终能被解读成句子 $\mathcal{Z}$ 当然是因为我是事先写了这个句子，然后转换成了 $\mathcal{Y}$，但其实 *chocolate cake* 换成任何一个两个单词组成的名词都讲得通（把 *mother* 换成 *father* 等等也讲得通），所以 $\mathcal{Y}$ 本质上并没有包含 *chocolate cake* 这个信息，而只包含了句子各部分间的关系。也就是说，如果没有其他渠道得来的信息，那么我们永远无法知道 EF、BCDEF、或其他单个字符究竟指的是什么（这就像为什么有些古老的语言会失传）。

第二，那些出现了多次的片段，只有一次是包含了非冗余信息。比如句子 $\mathcal{Z}$ 的第一部分明显可以压缩成 *my and your* [*mother made a chocolate cake*]（虽然语法有错误，但是意义是明确的），也就是说 [*mother made a chocolate cake*] 作为一个整体所包含的信息是没有变的，只是有了不同的限定词。

第三点很重要但是也很让人疑惑：这里我们能解读出那些没有重复的片段纯粹是因为我是事先有的句子 $\mathcal{Z}$，然后转换成 $\mathcal{Y}$。如果只看 $\mathcal{Y}$，是没有可能判断出 T、P 等只出现了一次的字符到底是噪音还是真含有信息。另外，我们将 T 写成 *my* 之后，能理解其是指"我的"的意思，也是因为 *my* 这个单词其实在我们脑子里重复过的。如果一个完全不会英文的人，他脑子里从来没有出现过 *my* 这个单词，那么即使你告诉他 $\mathcal{Z}$ 这句话，他也不能理解是"我的"的意思。所以根本上，一个字符、字符串、单词、短语等有意义、或者说带有信息，就必须有重复，要么在你的脑子里重复，要么在系统（即句子）本身重复，要么在别的什么地方。

[6]要有信息，那么也就是说，它不是完全随机出现的，言下之意就是，不是那么容易、随随便便就可以出现的，而是由某种机制产生的（物理机制、或者某种外星文明"刻意"为之都可以看作一种机制）。那什么样的字符串不是那么随机呢？重复，如果有重复的结构，那么就不是那么随机。也就是说，如果 $\mathcal{Y}$ 里有很多重复的字段，那么几乎可以肯定这些重复的字段是由某种特定的机制产生的，包含了一些信息，因为如果是完全随机的过程，产生很多重复的字段的概率是相当小的（其他没重复的字段有可能是噪声，当然也有可能是由另外的机制产生，故也可能是信息）。当然，这里面有个概率问题：如果重复的字段太短或者只重复了一次，那这种情况发生的概率也不会太小。但是，在本文中，我们暂时只考虑了只要重复（不论是短重复还是只重复一次）就表明不是完全随机的。

第四，上面的解读有一个很重要的假设，即我们把单个的字符看成了最基本的信息单元，即认为梯径中的基础集是由单个字符组成的。事实上，我们并没有充足的理由这么做：那些出现多次的字符（比如 R、H、K）看成基础组件的合理的，但是只出现一次的字符比如 P、S 是否能看作基础组件却有待商议。其实，我们完全可以把 PS 当成一个单一的组件，因为 P 和 S 并没有单独出现过。我们习惯于将 P、S 分开是因为在我们的语言中它们是分开的，但是如果在外星语言里，它们就是一个字符呢（从收到的信号来看，并没有充足的理由认为 P、S 是两个基础组件）?

所以，定义基础集的构成是第一步，这是一个相对主观的过程，可以根据一些假设、猜想，或其他一些事实来定义基础集，也可以只根据要处理的目标组件本身来做出定义。正如该外星信号 $\mathcal{Y}$ 的例子，可以像上文一样将单个字符看成基础组件，但也可以只取重复过的字符来组成基础集。如果是后者的话，$\mathcal{Y}$ 就应该看成 $\mathcal{Y}'$ =T [BCD[EF]] R [BCD[EF]] [T[EF]] H K R [EF] H K [MU] [T[EF]] [MU]，其最短梯径就是

$$J_{\mathcal{Y}'} = \{\mathrm{T}(2), \mathrm{R}(2), \mathrm{H}(2), \mathrm{K}(2), \mathrm{EF}(3), \mathrm{MU}(2), \mathrm{BCD} /\!/ \mathrm{TEF}, \mathrm{BCDEF}\}$$

其中，因为 B、C、D 总是一起出现的，所以将 BCD 看成了一个整体。故 $J_{\mathcal{Y}'}$ 的长度，亦即 $\mathcal{Y}'$ 的梯径度是 $\lambda(\mathcal{Y}') = 18 - 1 - 1 = 16$ 程，其中 18 是 $\mathcal{Y}'$ 的规模度，第一个 1 是 TEF 的规模度（因为 EF 是基础组件，所以只需 1 程以生成 TEF），第二个 1 是 BCDEF 的规模度（因为 BCD 和 EF 都是基础组件，所以也只需 1 程）。所以，也可以看出，主观选择的基础集不同，其梯径度也是不同的（但需要注意的是，一旦基础集定义好，在后续分析中就不应该改变）。至于到底应该选择哪一种基础集，是另外一回事，这取决于我们最终想要问什么问题，这也自然引出了下一节的讨论。

4.2 关于信息：人类语言（联合体系）

现将上面的句子 $\mathcal{Z}$ 放在人类语言（这里是英语）的体系中来看，这个时候，我们应该将什么当成基础组件呢，是单个字符、单个单词、还是重复的部分？最直观的答案是单个单词，因为单词是作为构成句子的基本单元。其实，将单个字符当成基础组件也是可以的，只是这种情况下，梯径会多一层，即，将字符组成单词的那一层。但原则上，这两种做法并不会产生本质区别，更像是同一个量采用了不同的单位。

然而，在英语的框架下，只将重复的部分（这里以重复的单词为例，当然这也适用于重复的字符）当成基础组件是不合适的。因为虽然这些单词在 $\mathcal{Z}$ 中没有重复过（比如 *not*），但是在整个英语的框架下是有重复的，也就是说，在其他可能的英语句子中，它是重复的。所以，在英语的框架下讨论问题，实际上是将所有可能的英语句子和目标句子看成一个整体上的“**联合体系**”：此时，即使在目标句子 $\mathcal{Z}$ 中没有重复的单词，在这个联合体系中也是重复的，所以必须将其考虑成基础组件。

第 4.1节中外星信号的例子 $\mathcal{Y}$ 是一个孤立体系，即，没有其他的信号、句子可以跟 $\mathcal{Y}$ 一起考虑成一个联合体系。所以，在 $\mathcal{Y}$ 中没有重复的字符，在整个孤立体系中（也就是这个句子本身）也是没有重复的，这也正是为什么我们没有强烈的理由将其当作基础组件。所以，对于孤立体系，我们有两种选择（都已在第 4.1节中提到）：一是做一些假设然后将单个字符作为基础组件，如 $J_{\mathcal{Y}}$；二是只将重复的字符作为基础组件，而未重复的字符当做噪声，如 $J_{\mathcal{Y}'}$。这两条路其实都对，也就是对原始字符串的两种解读，至于谁更能反应现实，则是另外一回事，需要从另外的角度探讨[7]（参考附录 E 关于如何运用孤立体系/联合体系的概念来寻找智慧生命的证据）。

[7]最后顺便说一下：假设在外星信号的例子中，我们收到了 $\mathcal{Y}$ 这个字符串 2 次、3 次甚至 100 次，那么应该怎么考虑呢？那么我们就几乎可以肯定（如果收到无穷多次，就是完全肯定），其中的所有字符都不是噪声。这种情况跟我们只收到 $\mathcal{Y}$ 一次，但是假定 $\mathcal{Y}$ 中没有噪声（即 $\mathcal{Y}$ 本身完完全全都是“外星人想要发送给我们的信息”）是等价的。这个时候，基础集就必须包含所有的字符。不过，我们仍然不能确定到底应该把 BCD 考虑成一个字符（像 $J_{\mathcal{Y}'}$ 中一样）还是 3 个字符（像 $J_{\mathcal{Y}}$ 中一样）。如果再多一个假想，即，我们还接收到了另外的一些字符串，在那些字符串中，B、C、D 单独出现过很多次（比如说收到很多 $\mathcal{Y}$ 中的字符顺序被打乱的字符串）。那这个时候，我们就可以将单个字符考虑成基础组件了，像 $J_{\mathcal{Y}}$ 一样。

关于第 4.1节的例子，有一点需要强调：如果我们收到很多条类似于 $\mathcal{Y}$ 的信号，应该怎么处理呢？这种情况下，我们必须将 $\mathcal{Y}$ 和其他收到的信号一起考虑为一个联合体系，然后分析这整个联合体系的梯径（显然地，其规模度很高）。若最后发现只有 BCDEF 是重复的且只重复了一次，那么其梯径度就会跟其规模度很接近、其有序度会非常低，意味着这个体系更有可能是随机的。更进一步，如果原则上有无穷多条信号而我们只收到了一条 $\mathcal{Y}$，又该如何处理呢？这种情况下就不得不接受这个事实：我们只能将 $\mathcal{Y}$ 考虑为一个孤立体系，这可能会导致错误的或者过于外推的结论（因此可能需要额外的证据，但那就是另一个问题了）。

现在我们可以回到第 1节所提出的那个问题：在某种语言中的一个句子所包含的“信息”实际上包含了两层意思。第一层是狭义上的信息，是指将这个句子放在它所属的语言之中，将它们考虑成联合体系之后，我们能从中提取出的“信息”，即梯径。第二层意思是在第一层意思的基础之上，即，将原来的句子用梯径描述之后，各个梯元所真正对应的客观事物、客观动作等到底是什么。第一层意思是关于句子本身（即“句法信息”），第二层意思是关于句子与客观世界的连接（即“语义信息”）。

4.3 关于：生命起源

对生命能够出现的概率到底有多低这个问题，有一个很著名的比喻“垃圾场龙卷风”[38]：假设有一堆废铜烂铁或者各种零部件散落在地上，生命出现的概率大概会和一阵龙卷风刮过使得这些零件组装成一架波音飞机的概率差不多。当然这只是一个形象的比喻，有助于让我们直觉上了解生命的出现到底有多难，但这个比喻容易产生很多误导。最大的问题是它假设了制造飞机的困难程度在于用最基本构件（比如螺丝、半导体、垃圾场里的小零件）组装飞机的难度，而这种假设其实是不正确的。下面我们来探讨这个问题。

一个人要设计一架飞机到底有多难？首先我们将这个问题极度简化，假设飞机只由发动机、螺旋桨、机翼、控制电路这四个部分构成。如果对于一个生活在公元前的人来说，他需要首先发明这四个部分，然后将它们组合起来。所以对于他来说，制造飞机的难度是发明这四部分的难度加上组装的难度。但是，一起发明这四个部分要比分别单独发明它们更简单，比如，他需要发明冶炼技术以获得金属，虽然这四个部分都需要金属，但是他只需要发明冶炼技术一次；他需要发现空气动力学以设计出机翼和螺旋桨的形状，但是他也只需要发现一次空气动力学；飞机上有两个机翼、螺旋桨、发动机，但是他也只需要发明这几个部件一次。

所以飞机设计的难度并不是各个部分独立的难度加起来，而是要扣除掉重复的部分，这是“制造飞机的难度比原来想象的简单”这个说法的第一层意思。

比“原来想象的简单”这个说法的第二层意思是基础集不同，即是说，如果这人生活在 20 世纪，那么这四个部分其实都已经是现成的技术。在梯径的概念下，对于这个现代人来说，他的基础集包括了发动机、螺旋桨、机翼、控制电路，那这时飞机的梯径度（也就是其难度）就是 4 程。而对于古人，他的基础集只包括矿石和石油，所以其设计飞机的难度就要远远高于 4 程。

也就是说，站在 20 世纪看，人类制造出飞机是相对很容易的，飞机出现的概率是很大的；而站在公元前来看，飞机出现的概率是非常非常小的。20 世纪的人所面对的基础集相当于公元前的人面对的基础集经过很多次生成操作、产生了很多更高层级的梯元之后的那个集合。

上面关于生命的比喻实际上是把生命类比成飞机，把物理世界的原子类比成散落在垃圾场里的零件。大量原子经过某种简单过程（比如龙卷风）组合成生命系统（比如细胞）的概率的确非常非常低。但是，一方面，生命本身并没有我们原来想象的那么复杂，因为很多部分都是重复的（下面会进一步讨论）；另一方面，生命的出现应该不是一蹴而就，而是类似于通过生成梯元的方式，使得系统慢慢变得复杂，最后出现生命也就理所当然了，就像 20 世纪出现飞机一样。

这里我们具体讨论第一方面，即生命并没有我们原来想象的那么复杂，因为有很多重复的部分。

首先，梯径对任何实体都是适用的，当然分子也不例外。但是对于分子，具体应该怎样定义生成操作、怎样定义梯径的单位长度，跟上文的字符串应该是不一样的。这里我提供一个合理的建议（也可能有其他策略，比如参见 [33] 的另一个对于分子的尝试，将分子想象成由分子键组成而不是原子）：

- 生成操作定义为将若干个分子结构或碎片（即组件）结合在一起，结合的方式是某些原子之间生成化学键。
- 梯径的单位长度定义为一个化学键的产生。也就是说，如果一次生成操作产生了 n 个化学键，那么该生成操作的长度就是 n 程。

注意，分子碎片或亚结构能够以很多种可能的方式组合，因此，生成操作可以有不同的类型（正如我们在第 2.2节定义生成操作之后已提到过的）。比如，CH_3CHCH_2、CH_3 和 H 可以组合成 $CH_3CH_2CH_2CH_3$（即丁烷）或 $HC(CH_3)_3$（即异丁烷）。具体选择了哪些生成操作需要作为梯径的必要附加信息而记录下来。

根据梯径的概念，对一群分子而言，同时生成他们的难度（即复杂度中由梯径度描述的那个方面）要比原来想像的小（“原本想像”指代通过组装单个原子来生成分子）。比如，NADH 中有两个核糖（ribose）基团，所以 NADH 的梯径度要比直接数它包含了多少个原子要小；RNA 的骨架都是由很多相同的核糖环和磷酸基团构成，很多蛋白质分子（比如 Tetrameric proteins）都是由很多重复的结构拼在一起的，所以它们的梯径度也要比其包含的原子数小得多；在基因组中基因序列也有很多重复的片段，所以基因组的梯径度也比碱基对的数目要小得多。还有一点就是，当我们将一堆分子一起考虑的时候（即，考虑目标系统而非单个目标组件的时候），其梯径度要比单独考虑它们的梯径度之和要小，比如整体考虑 1 *mol* NADH、1 *mol* ATP、1 *mol* FAD 这些分子的时候，它们很多部分都是重复的。

我们经常说蛋白质分子很复杂是因为我们觉得这个分子中包含了这么多原子为什么还“设计”得这么精巧 [39]，但其实当我们扣除蛋白质分子的重复结构之后，它的“复杂度”要小很多（更准确地说是它的梯径度比较小，相对于其规模度）。生命也是一样，当我们把生命系统（比如一个细胞）当成一个整体来看的时候，它的复杂度要比它每个部分复杂度相加之和小得多。虽然仍然可能很复杂，但是这会使得生命起源看起来比原来想像的容易得多。

4.4 关于：生命为什么是有序的

我们从一个观察开始讨论。在梯径系统中（即凡是满足“能够生成新组件”和“复制”的系统），每生成一个新梯元，整个系统的梯径度就增加了，相当于整个系统所蕴含的“信息”量就增加了；每复制一个梯元，相当于整个系统的有序度就增加了。所以，梯径系统的梯径度和有序度会自然而然地增加，即自然而然地朝着“复杂”的方向演化[8]：这是梯径系统的两个基本性质（即“能够生成新组件”和“复制”）的必然结果。

现在，在问“生命为什么是有序的”这个问题之前，我们问：如何理解“生命是有序的”（这是通过观察很容易得出的一个直觉的结论）？事实上，我们有两种方式来理解这里的“有序”二字。以一个细胞为例，第一种理解比较直观，即认为大量原子先组成了一些大分子、然后组织成细胞器、最终组织成细胞，而不是胡乱地将原子堆放在一起。

第二种理解是，构成该细胞的所有这些原子不能像理想气体那样弥散在该细胞所占有的空间内，而是被约束在了这些大分子以及细胞器的位置、能级等限制上（可以看出，这是在热力学意义下的

[8]这里提一下“时间之箭”这个著名的隐喻。在封闭系统中，时间之箭是熵增的方向，这也正是这个隐喻最初的意思。而从本文可以得出结论，在梯径系统（其必定是开放的）中，时间之箭应该是熵减的方向（在耗散系统中，其也必定开放，时间之箭也是熵减方向；而在开放的、非梯径、非耗散系统中，时间之箭的方向尚待深入研究）。但是也值得注意，“时间之箭”只是一个隐喻，它并不能解释是时间之箭导致了熵增、熵减，还是熵增、熵减导致了时间之箭；它实际上只是对一个现象的描述、隐喻而已。

解读）。所以这个细胞所包含的微观态数量比同等规模下理想气体的微观态数量少得多，即意味着，细胞的热力学熵很低（参考附录 F 中的关于梯径和香农熵的联系的更多讨论）。

第一种理解指的是“梯径意义下的有序”，第二种理解则是非常明确地指“在热力学熵的意义下有序”。可以看出，这两种意义下的“有序”并不是一回事，虽然都能在各自的意义下解释得通。

根据我们如何解释“有序”，我们对于“生命为何有序”这个问题也会有不同的角度。在梯径的意义下，这个问题似乎迎刃而解，因为梯径系统（生命属于梯径系统）朝着越来越有序的方向演化是它的基本特性，是其演化机制决定的[9]。而在热力学熵的意义下，这个问题仍有一些神秘之处，因为根据热力学第二定律，封闭系统的熵是自发增加的。虽然生命作为开放系统，其熵减并不违反热力学第二定律，但我们仍然不明白其中的机制，即，生命是怎样组织自己使得其可以抵抗热力学熵增（有些作者使用更笼统但听起来更有趣的说法 [40, 41]，比如“生命的熵是倾向于降低的”、“生命以负熵为食”等，但实际上这些说法都是等价的）。

如果“生命必然是一种梯径系统（也可以说，生命将自己组织成了梯径系统）”，并且“梯径系统可以抵抗热力学熵增”这两种说法是正确的，那么“生命是怎样组织自己使得其可以抵抗热力学熵增”这个问题就解决了。上文已经说明生命正是一种梯径系统，所以我们只需弄清楚梯径系统是否可以抵抗热力学熵增。我想这个答案是肯定的，但本文尚不能给出证明，这还需要更多的理论和实证研究。但至少我们将“生命为何有序”这类问题归结到了梯径系统的方向上，而梯径系统有着严格的数学定义，使得这类问题有了明确的方向。通过这些讨论中，我们希望传达的是：在研究诸如生命为何有序这类问题时，梯径可以提供一个全新的角度。

5 结论

本文试图说明，我们需要一种新方法来刻画某些客体（既非从统计分布中选出、亦非无限大比如无限长的字符串）的复杂性。然后我们提出了“梯径”方法，通过将有限大的客体解构成偏序多重集，来刻画其如何通过结合或修改已有组件的方式而被生成出来。从最短梯径中，我们能给出关于其所蕴含信息的两个测量，即梯径度和有序度，它们刻画了复杂性的两个不同方面：即，目标组件的生成难度、及其有序程度。该理论框架使得我们必须区分开孤立体系和联合体系，并且使得关于生命起源和进化的问题更加清晰明朗。这里我们仅仅讨论了梯径可能有的广泛应用的很少几个方面，我们希望这种方法在将来可以成功地运用在诸多领域，比如语言学、科技演化、和外太空生物学等。

说明：作者贡献、资助、致谢等信息请参见原文。本文开发的算法的源代码可以在此下载
https://github.com/yuernestliu/ladderpath

6 附录

附录 A：术语表

[9]事实上，梯径的层级关系很好地刻画了这种时间性、因果性：即，梯径的层级越多，表明该系统演化的时间性越久（注意，这并不一定代表演化的真实时间越长，就像在系统发生树里，进化分支只代表因果性，而不代表真实的时间）、不可逆转的因果性越深。

	名称:	参见:	定义及描述:
	基础集 (Basic set)	2.2节	包含构造目标客体的最基本元素的偏序多重集，记为 $\mathbf{S}_0$
	层 (Level)	2.2节	指代该偏序多重集中的偏序关系。在记法中，不同层由“//”隔开
⋆	组件 (Building block)	2.2节	该偏序多重集中的任意元素
	基础组件 (Basic building block)	2.2节	基础集 $\mathbf{S}_0$ 中的任意元素
	目标组件 (Target building block)	2.2节	最终生成的目标客体，记为 $\mathcal{X}$
⋆	生成操作 (Generation-operation)	2.2节	关于该偏序多重集的一种特殊操作，详见正文
⋆	梯径 (Ladderpath)	2.3节	一系列生成操作，其最终生成目标组件 $\mathcal{X}$。其可以被表示为一个偏序多重集，记为 $J_{\mathcal{X}}$。详见正文
⋆	梯元 (Ladderon)	2.3节	梯径中的任意组件
	平凡梯径 (Trivial ladderpath)	2.3节	一类特殊的梯径，其中所有的梯元都是基本组件
⋆	梯图 (Laddergraph)	2.3节	除了偏序多重集表示，一条梯径也可以被表示为一个图 (Graph，或网络)，这种表示称为梯图
⋆	梯径度 λ (Ladderpath-index)	2.4节	一个客体最短梯径的长度
⋆	规模度 S (Size-index)	2.4节	一个客体最短平凡梯径的长度
⋆	有序度 ω (Order-index)	2.4节	定义为 $(S-\lambda)$，参见 (4) 式
	梯径的长度单位	2.4节	一个跟具体系统有关的量（需用户定义）。在本文的字符串例子中，其定义为连接两个字符串一次这种操作。
	梯径的长度	2.4节	该梯径所对应的所有生成操作的长度之和
⋆	程 (*lift*)	2.4节	梯径的长度单位的名称
	最短梯径	2.4节	拥有最小长度的梯径
⋆	目标系统	2.6节	需要最终被一同生成的一组目标组件
⋆	梯径系统 (Ladderpath-system)	3节	满足以下两个条件的系统：(i) 具有生成新组件的能力，(ii) 一些或所有组件能够复制。
	孤立体系 (Isolated system)	4.1节	一个与其他任何体系都无关的体系（往往是一种假设）
	联合体系 (Non-isolated /united system)	4.2节	有时孤立某体系是不可能的。比如当考虑单一一个句子时，应把它放在其所属语言的框架下考虑，因此，该句子与其所属语言一同就是一个联合体系

注意，⋆ 表示重要的概念。

附录 B：计算序列的梯径的例子

为了更好地阐述，我们这里提供更多的例子演示如何构造一个目标组件的梯径（严格按照第2.2节和2.3节所说明的规则）。

首先我们举最平庸的例子，比如序列 $\mathcal{A}$ = ABCDEF。$\mathcal{A}$ 的其中一条梯径可以按例 B1 构造。为了不与正文中的集合 $\mathbf{S}_i$ 混淆，此附录中我们始终用 $\mathbf{R}_i$ 表示。$\mathcal{A}$ 的基础集为 $\mathbf{R}_0$ = { A(0), B(0), C(0), D(0), E(0), F(0) }。

例 B1. 第 1 次生成操作，$\mathbf{R}_0$: A+B+C+D+E+F=$\mathcal{A}$ → $\mathbf{R}_1$={ A(-1), B(-1), C(-1), D(-1), E(-1), F(-1) // $\mathcal{A}$(1) }

. 最后一次生成操作，$\mathbf{R}_1$: $\mathcal{A}$(-1) → $\mathbf{R}_2$={A(-1), B(-1), C(-1), D(-1), E(-1), F(-1) // $\mathcal{A}$(0)}

可以从 $\mathbf{R}_2$ 看出，例 B1 所对应的梯径为 $J_{\mathcal{A}} = \{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}, \mathrm{E}, \mathrm{F}\}$。事实上，它是 $\mathcal{A}$ 唯一的梯径。

第二个例子是序列 $\mathcal{B}$ = AAAAAAAA，其基础集为 $\mathbf{R}_0$ = { A(0) }，它的一条梯径可以如下构造：

例 B2. 第 1 次生成操作，$\mathbf{R}_0$: A+A=AA → $\mathbf{R}_1$={ A(-2) // AA(1) }

. 第 2 次，$\mathbf{R}_1$: AA+AA=AAAA → $\mathbf{R}_2$={ A(-2) // AA(-1) // AAAA(1) }

. 第 3 次，$\mathbf{R}_2$: AAAA+AAAA=$\mathcal{B}$ → $\mathbf{R}_3$={ A(-2) // AA(-1) // AAAA(-1) // $\mathcal{B}$(1) }

. 最后，$\mathbf{R}_3$: $\mathcal{B}$(-1) → $\mathbf{R}_4$={ A(-2) // AA(-1) // AAAA(-1) // $\mathcal{B}$(0) }

故对应于例 B2 的梯径为 $J_{\mathcal{B}} = \{\mathrm{A}(2) \,/\!/\, \mathrm{AA} \,/\!/\, \mathrm{AAAA}\}$。然后我们构造 $\mathcal{B}$ 的另一条梯径：

例 B3. 第 1 次生成操作，$\mathbf{R}_0$: A+A+A=AAA → $\mathbf{R}_1$={ A(-3) // AAA(1) }

. 第 2 次，$\mathbf{R}_1$: AAA+AAA=AAAAAA → $\mathbf{R}_2$={ A(-3) // AAA(-1) // AAAAAA(1) }

. 第 3 次，$\mathbf{R}_2$: A+A+AAAAAA=$\mathcal{B}$ → $\mathbf{R}_3$={ A(-5) // AAA(-1) // AAAAAA(0) // $\mathcal{B}$(1) }

. 最后，$\mathbf{R}_3$: $\mathcal{B}$(-1) → $\mathbf{R}_4$={ A(-5) // AAA(-1) // AAAAAA(0) // $\mathcal{B}$(0) }

故对应于例 B3 的梯径为 $J'_{\mathcal{B}} = \{\mathrm{A}(5) \,/\!/\, \mathrm{AAA}\}$。鉴于我们在第 2.4节已经介绍过梯径长度的概念，可知 $J_{\mathcal{B}}$ 的长度为 4 程，$J'_{\mathcal{B}}$ 长 6 程。事实上，$J_{\mathcal{B}}$ 是 $\mathcal{B}$ 的最短梯径（可以由第 2.7节介绍的算法来确认），因此 $\mathcal{B}$ 的梯径度为 $\lambda(\mathcal{B})$= 4 程。

另一个例子是序列 $\mathcal{C}$ = ATGTGCATG，其基础集为 $\mathbf{R}_0$ = { A(0), T(0), G(0), C(0) }，它的一条梯径可如下构造：

例 B4. 第 1 次生成操作，$\mathbf{R}_0$: T+G=TG → $\mathbf{R}_1$={ A(0), T(-1), G(-1), C(0) // TG(1) }

. 第 2 次，$\mathbf{R}_1$: A+TG=ATG → $\mathbf{R}_2$={ A(-1), T(-1), G(-1), C(0) // TG(0) // ATG(1) }

. 第 3 次，$\mathbf{R}_2$: ATG+TG+C+ATG=$\mathcal{B}$ → $\mathbf{R}_3$={ A(-1), T(-1), G(-1), C(-1) // TG(-1) // ATG(-1) // $\mathcal{C}$(1) }

. 最后，$\mathbf{R}_3$: $\mathcal{C}$(-1) → $\mathbf{R}_4$={ A(-1), T(-1), G(-1), C(-1) // TG(-1) // ATG(-1) // $\mathcal{C}$(0) }

故对应于 B4 的梯径为 $J_{\mathcal{C}} = \{\mathrm{A}, \mathrm{T}, \mathrm{G}, \mathrm{C} \,/\!/\, \mathrm{TG} \,/\!/\, \mathrm{ATG}\}$。当然，$\mathcal{C}$ 还有更多其他的梯径，这里不再展示。

再者，对序列 $\mathcal{K}$ = XYXYXYZXYY，其基础集为 $\mathbf{R}_0$ = { X(0), Y(0), Z(0)}，它的一条梯径可如下构造：

例 B5. 第 1 次生成操作，$\mathbf{R}_0$: X+Y=XY → $\mathbf{R}_1$={ X(-1), Y(-1), Z(0) // XY(1) }

. 第 2 次，$\mathbf{R}_1$: XY+XY+XY+Z+XY+Y=$\mathcal{K}$ → $\mathbf{R}_2$={ X(-1), Y(-2), Z(-1) // XY(-3) // $\mathcal{K}$(1) }

. 最后，$\mathbf{R}_2$: $\mathcal{K}$(-1) → $\mathbf{R}_3$={ X(-1), Y(-2), Z(-1) // XY(-3) // $\mathcal{K}$(0) }

故对应于例 B5 的梯径为 $J_{\mathcal{K}} = \{\mathrm{X}, \mathrm{Y}(2), \mathrm{Z} /\!\!/ \mathrm{XY}(3)\}$。

最后的例子是序列 $\mathcal{L} = \mathrm{GGGCAUCAUAUAUAUG}$，其基础集为 $\mathbf{R}_0$ = { A(0), U(0), G(0), C(0) }，它的一条梯径可如下构造：

例 B6. 第 1 次生成操作，$\mathbf{R}_0$: A+U=AU $\to$ $\mathbf{R}_1$={ A(-1), U(-1), G(0), C(0) /// AU(1) }

. 第 2 次，$\mathbf{R}_1$: C+AU=CAU $\to$ $\mathbf{R}_2$={ A(-1), U(-1), G(0), C(-1) /// AU(0) /// CAU(1) }

. 第 3 次，$\mathbf{R}_2$: G+G+G+CAU+CAU+AU+AU+AU+G=$\mathcal{L}$ $\to$ $\mathbf{R}_3$={ A(-1), U(-1), G(-4), C(-1) /// AU(-3) /// CAU(-1) /// $\mathcal{L}$(1) }

. 最后，$\mathbf{R}_3$: $\mathcal{L}$(-1) $\to$ $\mathbf{R}_4$={ A(-1), U(-1), G(-4), C(-1) /// AU(-3) /// CAU(-1) /// $\mathcal{L}$(0) }

故对应于例 B6 的梯径为 $J_{\mathcal{L}} = \{\mathrm{A}, \mathrm{U}, \mathrm{G}(4), \mathrm{C} /\!\!/ \mathrm{AU}(3) /\!\!/ \mathrm{CAU}\}$。

附录 C：计算石头图案的梯径

这里我们阐述如何计算第 2.5节中石头图案的梯径。以图案 [i] 和 [22] 为例，如图 B4a 和 B4b。基础集包含单个石头，生成操作包括平移、旋转一个组件并将其与另一个组件组合在一起。图中的展示严格地遵循梯径的定义（见第 2.3节）。

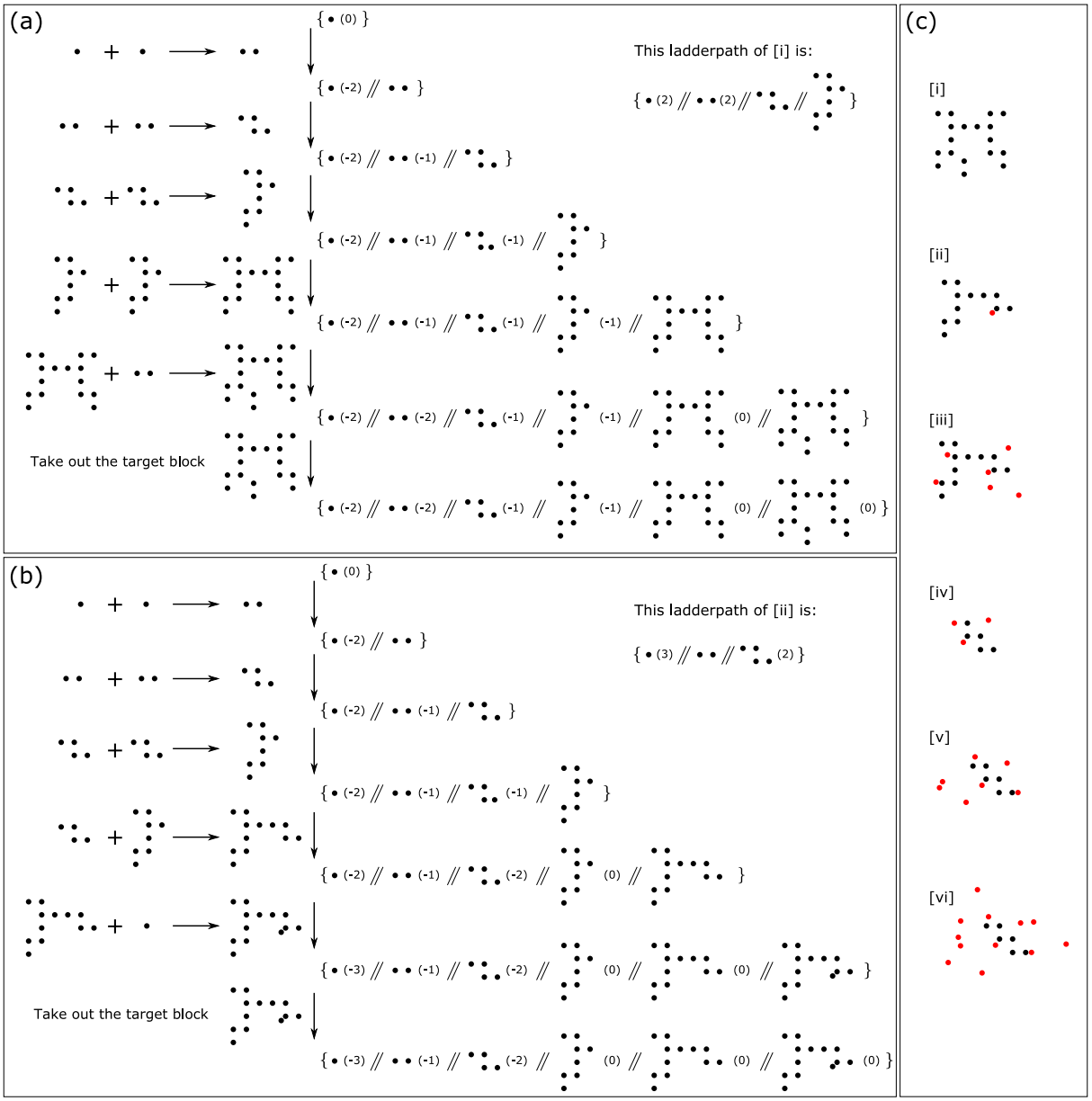


图 B4: **(a)** 最终生成图案 [i] 的一系列生成操作。左边的每一行代表一次生成操作。**(b)** 最终生成图案 [ii] 的一系列生成操作。**(c)** 展示了 6 个石头图案，其中高亮了不规则的部分。红色表示那些石头是无规则摆放的，因此那些石头必须一个一个添加。比如，对图案 [iii]，在黑色部分生成之后，需要 6 次更进一步的生成操作，而每一次就是添加一个“红色”石头。

附录 D：计算最短梯径的算法的例子

现以第 2.6节中的目标系统 $\mathcal{Q}$ = {ABDEDBED(2), ABDED, ABDABD, CAB(2), ED(3)} 为例阐述第 2.7节中描述的算法：

i. 首先构造一个空的多重集 $\mathcal{H}$ 以储存组件；最后梯径可以从 $\mathcal{H}$ 中方便地得到。

ii. 从目标系统 $\mathcal{Q}$ 开始，我们只在 $\mathcal{Q}$ 中保留每种不同组件的一个实例，并将所有其他的重复实例放入 $\mathcal{H}$。因此，我们有 $\mathcal{Q}$ = {ABDEDBED, ABDED, ABDABD, CAB, ED} 和 $\mathcal{H}$ = {ABDEDBED, CAB, ED(2)}（若组件后没有括号明确写出重数，则表明其重数为 1）。

iii. 然后是第 3 步，以事先确定的系统切割方案一直切割 $\mathcal{Q}$ 中的组件，直到 $\mathcal{Q}$ 中有至少 2 个子序列相同。可能有很多种系统切割方案，其中一种是：将切割位置用二进制数标号然后按顺序计数。比如对于字符串 $abcd$，可能的切割位置有 3 个，因此二进制数的顺序计数是 001、010、011、100、101、110、111，这些数可以被想象为一套完整的切割方案的序列，其中 1 代表切割，0 代表不切割。所以，$abcd$ 的切割方案为：$abc|d$、$ab|cd$、$ab|c|d$、$a|bcd$、$a|bc|d$、$a|b|cd$、$a|b|c|d$。在这个具体的例子中，当数到 0000100 时，即，将 ABDEDBED 切割成 ABDED 和 BED 时，我们发现此时 $\mathcal{Q}$ 中有 2 个 ABDED。所以，将其中 1 个 ABDED 放入 $\mathcal{H}$ 中，然后有 $\mathcal{Q}$ = {BED, ABDED, ABDABD, CAB, ED} 和 $\mathcal{H}$ = {ABDEDBED, CAB, ED(2), ABDED}。

iv. 重复第 3 步，我们可以将 BED 分割成 B 和 ED，且将 ABDED 分割成 ABD 和 ED。然后发现此时 $\mathcal{Q}$ 中一共有 3 个 ED 了。所以将其中 2 个 ED 放入 $\mathcal{H}$ 中，则有 $\mathcal{Q}$ = {B, ABD, ABDABD, CAB, ED} 和 $\mathcal{H}$ = {ABDEDBED, CAB, ED(4), ABDED}。

v. 重复第 3 步，我们可以再将 ABDABD 分割成 ABD 和 ABD。然后发现此时 $\mathcal{Q}$ 中一共有 3 个 ABD。所以将其中 2 个 ABD 放入 $\mathcal{H}$ 中，则有 $\mathcal{Q}$ = {B, ABD, CAB, ED} 和 $\mathcal{H}$ = {ABDEDBED, CAB, ED(4), ABDED, ABD(2)}。

vi. 重复第 3 步，我们可以再将 ABD 分割成 AB 和 D，且将 CAB 分割成 C 和 AB。然后发现此时 $\mathcal{Q}$ 中有 2 个 AB。所以将其中 1 个 AB 放入 $\mathcal{H}$ 中，则有 $\mathcal{Q}$ = {B, D, C, AB, ED} 和 $\mathcal{H}$ = {ABDEDBED, CAB, ED(4), ABDED, ABD(2), AB}。

vii. 重复第 3 步，我们可以再将 AB 分割成 A 和 B。然后发现此时 $\mathcal{Q}$ 中有 2 个 B。所以将其中 1 个 B 放入 $\mathcal{H}$ 中，则有 $\mathcal{Q}$ = {B, D, C, A, ED} 和 $\mathcal{H}$ = {ABDEDBED, CAB, ED(4), ABDED, ABD(2), AB, B}。

viii. 重复第 3 步，我们可以再将 ED 分割成 E 和 D。然后发现此时 $\mathcal{Q}$ 中有 2 个 D。所以将其中 1 个 D 放入 $\mathcal{H}$ 中，则有 $\mathcal{Q}$ = {B, D, C, A, E} 和 $\mathcal{H}$ = {ABDEDBED, CAB, ED(4), ABDED, ABD(2), AB, B, D}。

ix. 然后再也没有重复的字符或字符串了。现在把剩下的字符全部放入 $\mathcal{H}$ 中，得到 $\mathcal{H}$ = {ABDEDBED, CAB, ED(4), ABDED, ABD(2), AB, B(2), D(2), C, A, E}，而 $\mathcal{Q}$ 变成空集。

此时的 $\mathcal{H}$ 即记录了一条梯径。最后根据以上分割的方式，我们可以将 $\mathcal{H}$ 中各字符串的层次关系排列清楚，即得到偏序多重集 $\mathcal{H}$ = {A, B(2), C, D(2), E ⫽ AB, ED(4) ⫽ CAB, ABD(2) ⫽ ABDED ⫽ ABDEDBED}，即得到了一条梯径（即第 2.6节中的 (7) 式 $J_{\mathcal{Q}}$）。

附录 E：寻找智慧生命的证据

这里我们考察“寻找智慧生物的证据”这个问题，这实际上是从相反的方向来考察梯径演化问题，即，先知道了系统现在的状态，然后判断它是否是通过梯径的机制而到达这种状态的。我们这里提出 5 个假设命题 (hypothesis)：

1. 生命系统的有序度 ω 都很高，智慧生命、包括智慧生命所创造之物的有序度更高，且越有智慧，有序度越高（虽然智慧生命也可以创造出有序度低的东西，但这里考虑的是这个上限）。理由是，ω 描述了该系统到底跟完全随机产生的系统有多大的不同，即，ω 越大，跟随机系统越不同；而生命、智慧生命的参与一定是使系统偏离随机产生。

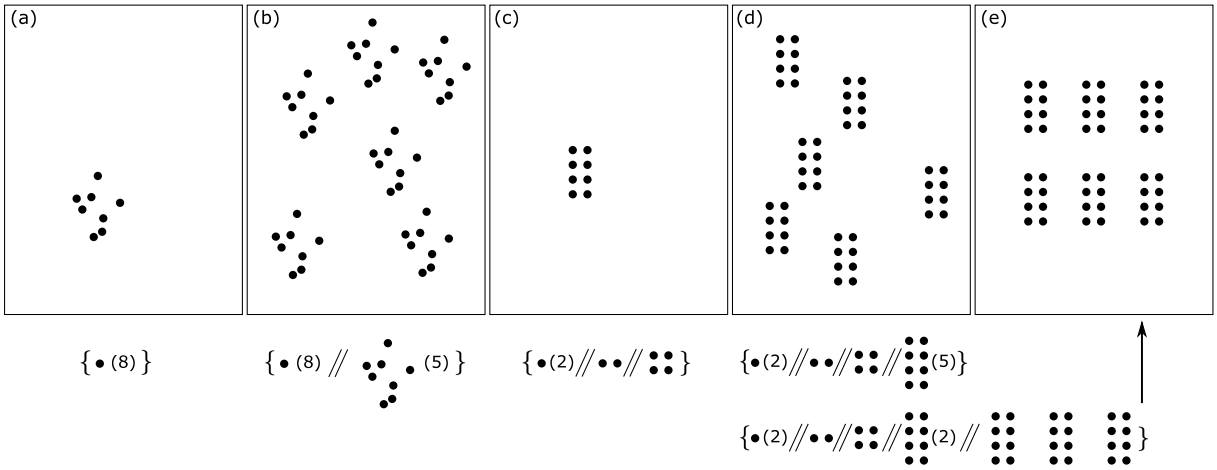
2. 但反过来，有序度 ω 高的系统并不一定都是生命、智慧生命、或智慧生命的造物（比如晶体，其由较简单的物理化学过程生成，非常有序却不是生命）。

3. 生命系统的梯径度 λ（相当于生成/重现该系统的代价和生成难度）也相对较高，智慧生命、包括智慧生命之造物的梯径度更高，且越有智慧，梯径度越高（同理，这里考虑的上限）。理由是，如正文中已述，梯径系统会自然而然地朝着越来越“复杂”（同时包括有序度和梯径度两个方面）的方向演化。

4. 但反过来，梯径度 λ 高的系统并不一定都是生命、智慧生命、或智慧生命的造物（比如随机的、混乱的气体分子系统，其梯径度也很高，却不是生命）。

5. 由上述 4 点可以推断出，如果一个系统是生命、智慧生命、或智慧生命的造物，那么它的有序度 ω 和梯径度 λ 会同时很高，且反之也成立（但至于多高才算是很高，则是另一个问题）。

我们通过一个例子来进一步阐述。如下图所示，设想系统 $(a)-(e)$ 是我们的宇宙飞船拍摄到的 5 个地外星球地表的清晰照片，其中每一个黑点代表一颗平凡无奇的石头，我们要通过这些照片判断是否这些星球上有生命、智慧生命、或这些石头图案是否是智慧生命所留下的遗迹。根据我们的假设命题，这 5 个系统里面最有可能的是 (b)，因为它的有序度和梯径度都同时很高；其次是 (d) 和 (e)，但 (e) 更有可能像是由物理过程产生的（比如结晶过程）。



	(a)	(b)	(c)	(d)	(e)
S	8	48	8	48	48
λ	8	13	4	9	7
ω	0	35	4	39	41

其中，S 为规模度，λ 为梯径度，ω 为有序度 $(:= S - \lambda)$。这三个量的单位都是“程”(*lift*)。

有两点需要说明：（1）并不是说 (b) 一定对应着生命或智慧生命，因为我们并不知道 ω 和 λ 同时多高才能对应着生命，它们只能给出一个相对概率。梯径理论本身并不能告诉我们生命或智慧生命所对应的 ω 和 λ 的阈值（如果真的存在阈值的话）。但是这个阈值的信息可能可以通过其他的途径得到，比如在地球上这个有生命和智慧生命的地方考察植物、动物系统的 ω 和 λ、原始部落的 ω 和 λ、城市人类的 ω 和 λ 等，通过这些数据来推测生命或智慧生命所对应的 ω 和 λ 的阈值（如果真的存在阈值）。（2）在比较有序度的时候，需要将基本组件设置成一致的才能进行比较，比如这里我们都是将一块石头作为基本组件。

上面的例子中，实际上相当于将每张照片看成一个孤立体系。现在我们考虑另外一个不同的例子，分别考虑下面 3 种情况：

(f) 我们在外星球表面上发现了一架飞机，记为 $\mathcal{F}$，跟地球上的空客飞机一模一样，甚至连窗户的位置都是一样的；

(g) 我们在一个外星球表面发现了一个看着像飞碟的东西，记为 $\mathcal{G}$，但我们完全不确定它到底是不是一个飞行器。它看着像一个圆盘，尺寸跟空客飞机差不多，表面有一些有规律的纹路，看着像有一个门，但是还不知道怎么进去。

(h) 在那个外星球上，我们发现了 2 个一模一样的 $\mathcal{G}$。

那么现在的问题是，哪一种情况更有可能有智慧生命参与?

在情况 (f) 中，如果把 $\mathcal{F}$ 看成一个孤立体系，那么它的梯径可以写成 {基础组件 $/\!/$ $\mathcal{F}_{rep}$}（其中 $\mathcal{F}_{rep}$ 是 $\mathcal{F}$ 中的重复部分），那么它的有序度就是 $\omega'_f = S(\mathcal{F}_{rep}) - 1 \doteq S(\mathcal{F}_{rep})$（根据(6)式）。但实际上我们应该把 $\mathcal{F}$ 跟地球上的空客飞机看成一个联合体系，因为 $\mathcal{F}$ 在我们的经验里是重复过的，所以 $\mathcal{F}$ 本身就应该是梯元，那么它的梯径就应该是 {基础组件 $/\!/$ $\mathcal{F}_{rep}$ $/\!/$ $\mathcal{F}$}（这大概相当于假设我们发现了 2 架一样的 $\mathcal{F}$），所以 (f) 的有序度就是 $\omega_f = S(\mathcal{F}_{rep}) + S(\mathcal{F})$。

在情况 (g) 中，我们只能把 $\mathcal{G}$ 当作一个孤立体系，因为我们不知道它到底是什么（故不能将其与地球上的飞机考虑为一个联合体系），所以它的梯径只能是 {基础组件 $/\!/$ $\mathcal{G}_{rep}$}（其中 $\mathcal{G}_{rep}$ 是 $\mathcal{G}$ 中的重复部分），则其有序度是 $\omega_g \doteq S(\mathcal{G}_{rep})$。在情况 (h) 中，虽然我们仍然不知道 $\mathcal{G}$ 到底是什么，但是由于 $\mathcal{G}$ 已经在我们考察的系统中重复了，所以它的梯径就是 {基础组件 $/\!/$ $\mathcal{G}_{rep}$ $/\!/$ $\mathcal{G}$}，其有序度是 $\omega_h \doteq S(\mathcal{G}_{rep}) + S(\mathcal{G})$。

如果我们进一步假设 $\mathcal{F}$ 和 $\mathcal{G}$ 的规模、以及其重复部分的规模都相同，即 $S(\mathcal{F}) = S(\mathcal{G})$ 且 $S(\mathcal{F}_{rep}) = S(\mathcal{G}_{rep})$，则有 $\omega_f = \omega_h > \omega_g$ 且 $\lambda_f = \lambda_h = \lambda_g$。那么综上，$(f)$ 和 (h) 的情况就非常相似，它们比情况 (g) 更有可能是智慧生命引起的。当然，如果 ω_g 和 λ_g 已经是很大的数值了，也就是说 $\mathcal{G}$ 的有序度和梯径度都已经很高了，若高出了我们以其他信息推断出的智慧生命的“阈值”，那么 (g) 当然也有可能是智慧生命引起的。

有理由相信，不仅生命的演化遵循梯径机制（如第 3节所述），文明的演化亦是如此（例如人类社会、科技的发展）。所以，我们可能是这样发现外星文明的：

1. 我们在某个星球上发现了一整套正在发生的梯径演化过程，或其留下的遗迹，但这些过程或者遗迹跟地球上所发生的完全不一样（比如可以是硅基生命、在液态甲烷中等等）。

2. 我们试图去找看起来跟我们的世界很像的、且梯径度 λ（即难度）很高的东西，比如飞机、有相似的特定复杂外形的建筑、相似的复杂代谢活动、相似的复杂机器，这其实相当于上面的情况 (f)。一旦找到与我们世界相似的高 λ 客体（那怕只有一个），就可以把它考虑成梯元（因为它在这个联合体系中重复了）。那么，整个联合体系的有序度和梯径度就都很高了，也就很可能是文明的产物（但如果我们和他们是真正独立的文明，那这种情况是不太可能的，因为从无数可能的文明中产生两个一模一样的文明的概率几乎是零）。

3. 我们只发现了一个遗迹或者机器，虽然只有一个样本，但是在这个样本里，有很多重复的、有层次结构的部分，最终算下来，它的有序度和梯径度都非常高。这种情况下，我们也几乎可以肯定这一定是外星文明留下的。

4. 当然不排除最后一种可能：我们发现了大量的梯径度 λ 非常高的物件，但是每个物件中却没有重复的部分，即每个物件的有序度非常低。这种情况我们也几乎可以肯定是外星文明。原因是，物件的数量很多，也就是说这个物件本身就是梯元，故整个系统的有序度其实是很高的；也因

为这个物件本身的梯径度很高，所以整个系统的梯径度也很高。只不过正如前文所述，这种文明发生的概率几乎是零（因为它并不符合梯径的描述），正如第 4.3节中那个垃圾场龙卷风的比喻。

附录 F：梯径和香农熵之间的联系

首先注意到，梯径和香农熵都只能描述一个客体或系统的句法信息，不能描述语义信息[10]。那它们之间还有什么其他联系呢？考虑一个例子，现设想一个目标系统 $\mathcal{D}$：一个封闭的容器中装了 $1mol$ 处于平衡态的丁烷（$CH_3CH_2CH_2CH_3$），即 6.022×10^{23} 个丁烷气体分子。我们知道梯径可以描述静态客体、结构、图案样式等。所以，这里我们能用梯径描述这 $1\ mol$ 丁烷分子的一个确定的微观态。这个微观态可以考虑成一个图案样式，在此样式中，气体分子全部被固定在随机的位置上，因而没有重复的样式（参见图 2b，可将那些石头想象成气体分子）。现在，若将“生成操作”定义为形成共价键，那么 $\mathcal{D}$ 的梯径就是：

$$J_{\mathcal{D}} = \{\mathrm{C}, \mathrm{H}(3) \mathbin{/\!/} \mathrm{CH_2} \mathbin{/\!/} \mathrm{CH_3CH_2} \mathbin{/\!/} \mathrm{CH_3CH_2CH_2CH_3}\ (6.022 \times 10^{23} - 1)\}$$

可以看出，$J_{\mathcal{D}}$ 主要描述的是丁烷的分子结构、与系统 $\mathcal{D}$ 是由 $1\ mol$ 随机分布的丁烷分子组成的这个事实。

另一方面，该情形下系统 $\mathcal{D}$ 的香农熵退化为热力学熵 [42]，可以由著名的 Sackur-Tetrode 公式算出，其是分子内能、分子数、和容器体积的函数 [43]。其可以理解成是由所有这些气体分子的运动、和分子内部的振动转动所约束的所有可能的微观态的数目（再取对数）。可以看出，这跟梯径所描述的是不一样的。

参考文献

[1] J. M. Smith, 20th century biology as a science of information (95/102), YouTube Video, youtube.com/watch?v=78ikxE5-POY (1997).

[2] A. Kolchinsky, D. H. Wolpert, Semantic information, autonomous agency and non-equilibrium statistical physics, Interface Focus 8 (6) (2018) 20180041. doi:10.1098/rsfs.2018.0041.

[3] J. P. Crutchfield, K. Young, Inferring statistical complexity, Phys. Rev. Lett. 63 (1989) 105–108. doi:10.1103/PhysRevLett.63.105.

[4] H. Thorén, P. Gerlee, Weak emergence and complexity, in: H. Fellerman, M. Dörr, M. Hanczy, L. Ladegaard Laursen, S. Mauer, D. Merkle, P.-A. Monnard, K. Støy, S. Rasmussen (Eds.), Artificial Life XII Proceedings of the Twelfth International Conference on the Synthesis and Simulation of Living Systems, MIT Press, United States, 2010, pp. 879–886.

[5] A. N. Kolmogorov, Three approaches to the quantitative definition of information, Problemy Peredachi Informatsii [Problems of Information Transmission] 1 (1965) 3–11. doi:10.1080/00207166808803030.

[6] G. J. Chaitin, On the length of programs for computing finite binary sequences, J. ACM 13 (1966) 547–569. doi:10.1145/321356.321363.

[10] 正如前文已经提到，一句话所包含的信息其实包含着两层意思：一是将句子本身与它所属的语言考虑成一个联合体系，我们能从中提取的“信息”，即梯径；二是在第一层意思的基础上，各个“信息单元”（若在梯径中，则是梯元）是怎样对应客观世界中的客体的。事实上，不管是梯径还是香农熵、热力学熵都不涉及第二层意思（它需要原句子之外的额外事实：比如“橙子”是指那种黄色的、可以吃的、富含维生素的、长成那种样子的水果）；而梯径所描述的第一层意思是最符合我们直觉上的“语义上的信息”的（和香农熵、热力学熵所指代的不太一样）。所以，我们可能更应该用梯径来描述一句话、一段文字等所包含的信息。

[7] G. J. Chaitin, On the length of programs for computing finite binary sequences: statistical considerations, J. ACM 16 (1) (1969) 145–159. doi:10.1145/321495.321506.

[8] G. J. Chaitin, Information-Theoretic Incompleteness, World Scientific, 1992. doi:10.1142/1861.

[9] A. K. Zvonkin, L. A. Levin, The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms, Russian Mathematical Surveys 25 (6) (1970) 83–124. doi:10.1070/RM1970v025n06ABEH001269.

[10] P. M. B. Vitányi, How incomputable is kolmogorov complexity?, Entropy 22 (4) (2020) 408. doi:10.3390/e22040408.

[11] A. Lempel, J. Ziv, On the complexity of finite sequences, IEEE Transactions on Information Theory 22 (1) (1976) 75–81. doi:10.1109/TIT.1976.1055501.

[12] J. Ziv, A. Lempel, A universal algorithm for sequential data compression, IEEE Transactions on Information Theory 23 (3) (1977) 337–343. doi:10.1109/TIT.1977.1055714.

[13] J. Ziv, A. Lempel, Compression of individual sequences via variable-rate coding, IEEE Transactions on Information Theory 24 (5) (1978) 530–536. doi:10.1109/TIT.1978.1055934.

[14] C. Adami, N. J. Cerf, Physical complexity of symbolic sequences, Physica D: Nonlinear Phenomena 137 (1) (2000) 62–69. doi:https://doi.org/10.1016/S0167-2789(99)00179-7.

[15] C. E. Shannon, A mathematical theory of communication, The Bell System Technical Journal 27 (3) (1948) 379–423. doi:10.1002/j.1538-7305.1948.tb01338.x.

[16] P. Grassberger, Towards a quantitative theory of self-generated complexity, International Journal of Theoretical Physics 25 (1986) 907–938. doi:10.1007/BF00668821.

[17] A. Achille, G. Paolini, G. Mbeng, S. Soatto, The information complexity of learning tasks, their structure and their distance, Information and Inference: A Journal of the IMA 10 (1) (2021) 51–72. doi:10.1093/imaiai/iaaa033.
URL https://doi.org/10.1093/imaiai/iaaa033

[18] M. Makowski, E. W. Piotrowski, P. Frąckiewicz, M. Szopa, Transactional interpretation for the principle of minimum fisher information, Entropy 23 (11) (2021). doi:10.3390/e23111464.
URL https://www.mdpi.com/1099-4300/23/11/1464

[19] M. Li, P. Vitanyi, An Introduction to Kolmogorov Complexity and Its Applications, 3rd Edition, Springer Verlag, 2008.

[20] G. Hansel, D. Perrin, I. Simon, Compression and entropy, in: A. Finkel, M. Jantzen (Eds.), STACS 92, Springer Berlin Heidelberg, Berlin, Heidelberg, 1992, pp. 513–528.

[21] J. Ziv, N. Merhav, A measure of relative entropy between individual sequences with application to universal classification, IEEE Transactions on Information Theory 39 (4) (1993) 1270–1279. doi:10.1109/18.243444.

[22] F. Jacob, Evolution and tinkering, Science 196 (4295) (1977) 1161–1166. doi:10.1126/science.860134.

[23] M. Middendorf, E. Ziv, C. H. Wiggins, Inferring network mechanisms: The drosophila melanogaster protein interaction network, Proceedings of the National Academy of Sciences 102 (9) (2005) 3192–3197. doi:10.1073/pnas.0409515102.
URL https://doi.org/10.1073/pnas.0409515102

[24] G. P. Wagner, M. Pavlicev, J. M. Cheverud, The road to modularity, Nature Reviews Genetics 8 (12) (2007) 921–931. doi:10.1038/nrg2267.
URL https://doi.org/10.1038/nrg2267

[25] S. Valverde, R. Sole, Hierarchical small worlds in software architecture, in: Software Engineering and Complex Networks, Vol. 14 of Dynamics of Continuous, Discrete and Impulsive Systems Series B, 2007, pp. 1–11.

[26] S. Valverde, Breakdown of modularity in complex networks, Frontiers in Physiology 8 (2017). doi:10.3389/fphys.2017.00497.
URL https://www.frontiersin.org/article/10.3389/fphys.2017.00497

[27] R. Solé, S. Valverde, Evolving complexity: how tinkering shapes cells, software and ecological networks, Philosophical Transactions of the Royal Society B: Biological Sciences 375 (1796) (2020) 20190325. doi:10.1098/rstb.2019.0325.
URL https://doi.org/10.1098/rstb.2019.0325

[28] D. Knuth, Evaluation of Powers, 3rd Edition, Addison-Wesley Professional, 1997, Ch. 4.6.3, pp. 461–485.

[29] S. M. Marshall, A. R. G. Murray, L. Cronin, A probabilistic framework for identifying biosignatures using pathway complexity, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 375 (2109) (2017) 20160342. doi:10.1098/rsta.2016.0342.

[30] A. Murray, S. Marshall, L. Cronin, Defining pathway assembly and exploring its applications, arXiv (2018). doi:10.48550/arXiv.1804.06972.

[31] S. M. Marshall, D. Moore, A. R. G. Murray, S. I. Walker, L. Cronin, Quantifying the pathways to life using assembly spaces, arXiv (2019). doi:10.48550/arXiv.1907.04649.

[32] S. M. Marshall, C. Mathis, E. Carrick, G. Keenan, G. J. T. Cooper, H. Graham, M. Craven, P. S. Gromski, D. G. Moore, S. I. Walker, L. Cronin, Identifying molecules as biosignatures with assembly theory and mass spectrometry, Nature Communications 12 (1) (2021) 3033. doi:10.1038/s41467-021-23258-x.
URL https://doi.org/10.1038/s41467-021-23258-x

[33] Y. Liu, C. Mathis, M. D. Bajczyk, S. M. Marshall, L. Wilbraham, L. Cronin, Exploring and mapping chemical space with molecular assembly trees, Science Advances 7 (39) (2021) eabj2465. doi:10.1126/sciadv.abj2465.

[34] P. Downey, B. Leong, R. Sethi, Computing sequences with addition chains, SIAM Journal on Computing 10 (3) (1981) 638–646. doi:10.1137/0210047.

[35] W. Hordijk, M. Steel, Detecting autocatalytic, self-sustaining sets in chemical reaction systems, Journal of Theoretical Biology 227 (4) (2004) 451–461. doi:10.1016/j.jtbi.2003.11.020.

[36] Y. Liu, D. J. T. Sumpter, Mathematical modeling reveals spontaneous emergence of self-replication in chemical reaction systems, Journal of Biological Chemistry 293 (49) (2018) 18854–18863. doi:10.1074/jbc.RA118.003795.

[37] Y. Liu, On the definition of a self-sustaining chemical reaction system and its role in heredity., Biology Direct 15 (15) (2020). doi:10.1186/s13062-020-00269-0.

[38] D. Gatherer, Finite universe of discourse: the systems biology of walter elsasser (1904-1991), The Open Biology Journal 1 (2008) 9–20. doi:10.2174/1874196700801010009.

[39] E. Zimmerman, A. Yonath, Biological implications of the ribosome's stunning stereochemistry, ChemBioChem 10 (1) (2009) 63–72. doi:10.1002/cbic.200800554.

[40] E. Schrödinger, What Is Life? The Physical Aspect of the Living Cell., Cambridge University Press, Cambridge., 1944.

[41] K. Wu, Q. Nan, T. Wu, Philosophical analysis of the meaning and nature of entropy and negative entropy theories, Complexity (2020). doi:10.1155/2020/8769060.

[42] X. Gao, E. Gallicchio, A. E. Roitberg, The generalized boltzmann distribution is the only distribution in which the gibbs-shannon entropy equals the thermodynamic entropy, The Journal of Chemical Physics 151 (3) (2019) 034113. doi:10.1063/1.5111333.

[43] D. V. Schroeder, An Introduction to Thermal Physics, Addison Wesley Longman, 1999.