计 算 机 研 究 与 发 展
Journal of Computer Research and Development
https://crad.ict.ac.cn
63(6): 1377−1389, 2026

观 点

计算范式的扩展与跃迁: 从 CS1.0 到 CS2+

李国杰
(中国科学院计算技术研究所 北京 100190)
(lig@ict.ac.cn)

Expansion and Transition of Computing Paradigms: From CS1.0 to CS2+

Li Guojie
(*Institute of Computing Technology*, *Chinese Academy of Sciences*, *Beijing* 100190)

Abstract This paper discusses the expansion and transition of computing paradigms. Firstly, it expounds on the great historical significance and limitations of Turing computation, defines what computing paradigm is, distinguishes between four types of computing paradigms: CS1.0, CS1.x, CS2.x and CS2+, and points out that computing paradigms evolve from "discrete, algorithm-based, rule-fixed, externally-assigned semantics, and closed monomers" to "continuous, dynamical systems, rule generation, endogenous semantics, and open ecosystems" in multiple dimensions, moving towards open evolutionary computation. Next, it is emphasized that the distinguishing criterion between CS1.x and CS2.x is the rule closure and semantic closure; It introduces typical cases of open evolutionary computing paradigms, including generative AI, structural computing, and institutional computing; emphasizes that the core issue of open evolutionary computation is governability. Finally, it discusses the theoretical basis and key technologies of open evolutionary computation, clarifies the significant importance of conducting basic research on open evolutionary computation, and calls on academia and industry to pay attention to the transition of this technological paradigm.

Key words computing paradigm; computer science 1.x (CS1.x); computer science 2.x (CS2.x); open evolutionary computation; rule closure; semantic endogeneity; dynamical system; structural computation; institutional computation; governability

摘 要 本文讨论计算范式的扩展与跃迁。首先阐述了图灵计算的伟大历史意义与局限性,定义了什么是计算范式,区分了4类计算范式: CS1.0、CS1.x、CS2.x和CS2+,指出计算范式是从"离散、算法、规则固定、外赋语义、封闭单体"向"连续、动力系统、规则生成、语义内生、开放生态"多维方向扩展,迈向开放演化计算。接着重点解释了CS1.x和CS2.x的区分判据是规则闭包和语义闭包;介绍了开放演化计算范式的典型案例,包括生成式AI、结构性计算、制度性计算等;强调开放演化计算的核心问题是可治理。最后讨论了开放演化计算的理论基础和关键技术,阐明开展开放演化计算基础研究的重大意义,呼吁学术界和产业界重视这一技术范式的变迁。

关键词 计算范式;计算机科学1.x;计算机科学2.x;开放演化计算;规则闭包;语义内生;动力系统;结构性计算;制度性计算;可治理

中图法分类号 TP301

DOI: 10.7544/issn1000-1239.202660189 **CSTR:** 32373.14.issn1000-1239.202660189

过去的80年中,以图灵机模型为理论基石、以冯·诺依曼架构为工程实现蓝图的计算机科学技术范式,构建了数字时代的基础设施,这一范式催生了电子计算机、集成电路、互联网、云计算与人工智能,

收稿日期: 2026-03-30; **修回日期**: 2026-04-23

其成就永载史册。

21世纪第3个十年,人类正站在计算文明的分水岭上。计算已不局限于函数映射,物理结构和社会制度开始成为计算对象,计算系统可能不终止地持续运行,动力系统等开放演化计算范式正在形成。新的计算范式不是对图灵范式的否定,而是对其适用边界的清醒认知与扩展。长期以来计算机界主要关注基于符号和算法的计算,开放演化计算的兴起是涉及人类认知和文明进程的计算范式变迁。我们发表这篇文章的目的,是呼吁学术界和产业界重视这一技术范式的变迁。

1 图灵计算的伟大历史意义与局限性

图灵计算模型是划时代的伟大科学成就,目前广泛使用的CPU、操作系统、编译器、软件工程方法等,本质上仍然运行在“图灵—冯·诺依曼”范式之中。图灵计算模型的3项奠基性贡献是:1)将“计算”从心理活动转化为“符号在规则下的机械操作”,给出了“什么是计算”的可操作定义,这是人类第一次拥有一个与人类能力无关的计算定义,使得可计算性和算法复杂性成为严肃的科学对象。2)建立了“可验证执行”的理论基础,让“正确执行”成为可证明、可测试、可复制的事实,直接催生了程序正确性等计算机科学技术。图灵计算是可验证性最高、工程可靠性最强、理论结构最完整的计算范式。3)计算与物理实现解耦,使得同一算法可在不同机器运行,计算成为“可工业化的知识形态”。

为定义“有效可计算性”,图灵1936年提出的a-machine基础模型(专用图灵机模型)有4条核心约束:1)状态空间封闭,系统的状态变化由有限规则完全刻画;2)固定的状态转移规则在运行过程中不能修改,规则完全外生,不能自主生成新规则;3)纯句法操作,目标外赋且可验证,语义由系统外的人赋予;4)被动执行,没有持续的环境交互,没有动态演化能力。设定这些约束条件是因为,只有规则固定,才能保证计算过程是机械的、无歧义的、可重复的过程。但图灵在同一篇文章中还提出了通用图灵机(UTM)模型,明确赋予了图灵机动态修改规则的能力。通用图灵机模型只定义了“如何操作符号”,没有定义“符号的意义从哪里来”,也没有强调“语义必须由外部人类赋予”。在他这篇经典文章中,图灵还提出了选择机(c-machine),选择机天生就是开放的、交互式的,可以在运行过程中持续接收外部输入、可以不停机。

图灵提出图灵机模型的目的,不是为了制造只能做固定数值计算的机器,而是为了制造能够思考、能够学习、能够拥有智能的机器。图灵机模型是一台能模拟所有可计算过程的通用机器,至今没有任何物理上可实现的计算机的能力超过通用图灵机。

但是,由于技术条件的限制,在过去70年里,计算机主要用来做专用自动机(a-machine)能做的事情,学术界和产业界把专用自动机的“规则固定”等约束,错误地推广到了通用图灵机上,形成了“所有计算机程序都必须预先写好、规则和语义不能内生”的固化思维,误以为这就是图灵计算模型的全部能力。这种坐井观天的思维惯性造成自缚手足的计算范式成为当今的主流计算范式,这一范式也划定了传统计算机科学的研究范围,本文称为CS1.0。为了区别于图灵模型,本文将流行了70年的计算范式称为“图灵范式”。撰写本文的动机就是要澄清模型和范式的区别。图灵模型的边界是可计算性的终极边界:任何可机械执行的过程,都可以用图灵机模拟。我们不必在突破图灵模型的可计算边界上花太多心思,而是要突破规则固定、语义外赋、封闭系统等图灵范式的束缚。这些束缚不是图灵模型本身的问题,而是后人添加的图灵范式的人为限制。

近年来出现的许多计算过程已突破了图灵范式的边界,例如在线强化学习中持续更新策略和目标函数,执行型智能体在开放环境中运行,神经形态硬件采用连续动力学进行计算,全球算力网络在运行中动态调度资源等。这些现象提示:计算范式正在从“封闭函数映射”扩展至“开放系统中的持续演化过程”。

图灵计算范式的基本假设是表示与计算可分离,但AI大模型的内部表示是高维连续流形结构,没有稳定、可命名的语义对象,表示就是状态,状态演化就是计算。表示与计算开始纠缠,计算不再是“对表示的操作”,而是“表示本身的演化”。演化没有“步骤”这一基本单位,其实现过程上不同于经典算法,这已经不是经典意义上的“信息处理”。

未来相当长的时间内,图灵计算范式都不会过时,基于图灵计算范式的CS1.0将继续担当计算机科研的主流范式。但CS1.0不是所有信息处理的唯一形式,更不是所有智能系统的终极框架。图灵计算范式的力量正是源于它刻意的简化和封闭,通过隔离环境、忽略噪声、设定规则、终态语义,将复杂问题转化为可形式化验证的数学对象,从而获得通用

性和抽象能力，使图灵模型可跨越技术代际，适用于从真空管到量子芯片的各种器件。图灵的伟大，不在于他用简单的模型抽象了各行各业的计算任务，而在于他精确划定了“机械计算”的边界。图灵计算范式的封闭性，如同地图的边界，它没有描绘海洋之外的世界，却让我们知道船该从哪里启航。

探索新计算范式的使命是构建智能时代的计算理论和技术框架。真实世界的计算不是一次性求解问题，而是在开放、动态、不确定环境中持续进行的意义建构、适应与行动，实现从理解世界、参与世界到塑造世界的跃升，让计算理论从“机器能不能算”和“计算有多难”走向“系统在开放环境中如何演化”。这正是当前人工智能、具身认知、复杂系统、人机协同等前沿领域共同指向的方向。

2 计算范式的演变和分类

2.1 什么是计算范式（computing paradigm）？

任何计算都可以抽象为问题空间、规则系统、资源约束和组织结构，问题的可求解性和计算系统的能力可以做 3 种不同维度的抽象：可计算性、计算复杂性和计算构成性。计算构成性是指计算在现实中由什么来实现，也称为计算范式。这 3 个维度分别针对上述不同的计算抽象部分，不是从低到高的分层，而是 3 条不同的正交轴。可计算性回答的是一个极其抽象的问题：对于任何有限描述、离散输入输出的问题，是否存在一个有限规则系统，能在有限步骤内保证给出正确答案？其本质是衡量一个封闭的规则系统对问题空间的穷尽能力，即“封闭性规则的极限”。可计算性是存在性维度，回答存不存在算法这一最底层的问题。计算复杂性是效率维度，回答实现算法需要多少资源这一计算成本问题。计算范式是结构性维度，回答问题如何表示、计算如何进行、规则如何生成、语义是否内生、系统是否开放等问题，强调“计算是如何构成的”，与可计算性不在同一维度。

定义 1. 计算范式。计算范式是指一类计算系统在信息表示方式（R）、状态演化机制（M）、规则生成机制（G）、语义生成机制（S）、系统边界条件（B）上所共享的结构性约束与构成原则，可表示为计算系统五元组（R, M, G, S, B）的等价类结构。

上述定义构成计算范式的 5 维模型。根据这一定义，不同计算范式就是由这 5 维坐标区分的等价类。例如，图灵范式就是满足五元组（离散符号表示，算法执行，外赋固定规则，外赋终态语义，封闭系统）的各种计算构成的等价类。表 1 显示了计算范式的演变方向。

把不同的计算系统放在这个 5 维空间中，可明显看出它们的区别，见表 2。计算范式向着类似人脑和人类社会的方向演化，如果把人脑和社会也看成计算范式，它们在 5 维范式空间的位置是（连续，动力，可变，内生，开放）。

Table 1 Evolution Direction of Computing Paradigm

表 1 计算范式演变方向

维度	图灵计算范式	计算范式演变方向
信息表示方式	离散符号表示	连续结构表示
状态演化机制	算法执行	动力系统演化
规则生成机制	外赋固定规则	规则可自生成
语义生成机制	外赋语义	语义内生
系统边界条件	封闭系统	开放生态

Table 2 The Coordinate Positions of Different Computations in a Five-Dimensional Paradigm Space

表 2 不同计算在 5 维范式空间中的坐标位置

系统	表示	机制	规则	语义	环境
传统软件	离散	算法	固定	外赋	封闭
RRAM 存内计算	连续	动力系统	固定	外赋	封闭
神经网络训练	连续	动力系统	半固定	外赋	半开放
LLM+Agent	连续	动力系统	可变	部分内生	开放
人脑	连续	动力系统	可变	内生	开放
人类社会	连续	动力系统	自生成	强内生	开放

总体而言，计算范式是从“离散、算法、规则固定、外赋语义、封闭单体”向“连续、动力系统、规则生成、语义内生、开放生态”多维方向扩张。计算范式演变没有改变“可计算问题集合”（图灵可计算性），而是扩展计算的表示方式、问题类型、结构组织这些“能力空间”，扩大了“可自然表达 + 可稳定运行 + 可工程实现”的问题区域。讨论计算范式时，我们重点关注的是“计算组织自由度”和“可治理性”。

CS1.0 的三大特征是终止性、静态语义和环境封闭性，在 AI 时代，这些特征全都失效，我们必须拥抱非终止、过程语义与开放演化。在许多新出现的计算中，计算对象不再是函数，而是状态演化轨迹；代码不再是静态指令，而是支配状态转移的微分/差分方程；计算执行的正确性不只是看最终结果，而是看轨迹属性，例如自动驾驶系统的目标不是“从 A 到 B”，而是“在所有可能扰动下，轨迹始终位于安全范围内”。我们把从经典的图灵计算扩展出来的计算

范式统称为“开放演化计算范式”。图灵计算范式将世界简化为可解问题，但实际上世界本身是一个未完成的过程。开放演化计算不承诺终极答案，而是提供在不确定性中航行的罗盘；不是问“输出是什么”，而要问“轨迹通向何方”。我们正从“计算的工匠”转变为“复杂演化的领航人”。

2.2 区分计算范式 CS1.0、CS1.x、CS2.x 和 CS2+

自图灵计算模型被提出以来，可计算问题的边界一直没有突破，在物理可实现范围内，迄今没有证据表明存在真正超越图灵可计算性的物理计算。当前的机器学习大模型、标准量子计算、脉冲神经形态芯片等，因其行为可被图灵机模拟，仍属于图灵可计算范畴。因此，计算机界有不少学者认为，只要没有出现突破可计算性边界的超图灵机，计算机领域所有的科研都属于 CS1.0，不必考虑计算范式扩展和变迁引发的计算机科学研究范围的扩张。这种观点可能出自对可计算性的误解，不利于计算机科学的与时俱进。可计算性只是刻画计算能力的最基础的维度，在图灵可计算的边界之内，计算范式已经发生明显变化。

实际上，CS1.0 框架已难以解释和指导新的计算。AI 大模型能生成代码、发现新知识、提出新猜想，已无法用“执行固定程序”的经典范式解释。具身智能体在开放环境中持续学习，其目标、技能、交互协议不断演化，远超“程序=全部行为”的假设。数字果蝇的复杂连接结构本身就是“算法”和“程序”。若仍用 CS1.0 语言来描述这些系统，会陷入概念失配，就如同用牛顿力学描述量子现象。现在的 AI 突破大多由跨学科团队（神经科学、控制论、生命科学等）驱动，涌现和世界建模已不再局限于“高效计算”，而是涉及认知建模、社会规范、系统演化和治理。计算机科学若固守“符号+算法”传统，将失去对智能时代核心议题的话语权，可能被边缘化为“工程支撑学科”。

根据本文的定义，一种计算范式是计算结构的 5 维空间中一个对应的小区域。云计算、区块链、存算一体计算、类脑计算、光计算、热力学计算、分子计算、量子计算等，都可以认为是新的计算范式。计算技术在不同维度的演变有不同的表现，有些技术演变是渐进的，只有程度上的模糊变化，没有明确的界限。技术变迁的影响也有大有小，不是每个维度的演进都称得上范式变迁。

我们可以根据不同维度的变化细分计算范式，例如：表示方式的变化称为 CS1.1，计算机制的变化称为 CS1.2 等。但是，细分计算范式并没有太大的实际意义。区分计算范式的最大意义是为计算机系统特别是智能系统的治理提供理论指导。5 个维度的治理难度呈现出一种递进的结构关系，其顺序大体上是：表示方式 → 计算机制 → 系统开放性 → 规则生成 → 语义生成。前两者主要决定系统如何表征与演化，后三者决定系统是否进入开放环境，如何在运行中生成规则与意义。就治理而言，真正的断裂点不在开放性，而在规则生成和语义生成，尤其是最终评价权的内生化。

规则是系统中用于引导或约束状态转移、动作选择、任务分解等行为过程的可执行因果机制。语义是系统中用于解释状态、事件或符号之意义差异的操作性理解框架。为了对计算范式做分类，我们需要先定义 2 个概念：规则闭包和语义闭包。

定义 2. 规则闭包。若存在一个在运行前给定的有限可描述的规则空间，运行中不会产生指定空间外的可执行规则类型，则称该系统的规则闭包成立。

定义 3. 语义闭包。若存在一个运行前给定的外部解释/评价框架，使得系统在运行中的一切状态、结果与过程都由其界定，不在框架外生成操作性意义结构来支撑后续运行，则称该系统的语义闭包成立。

规则闭包和语义闭包可以作为计算范式大分类的判据。

定义 4. CS1.x 是规则闭包与语义闭包同时成立的计算范式。

定义 5. CS2.x 是规则闭包或语义闭包至少一项失效的计算范式。

规则闭包和语义闭包的含义不是指机器完全不能生成规则和语义；而是指机器即使内生规则或语义，这些内生结果必须落在人类预先指定的规则和评价框架的闭包内。

每一种范式都应有与之对应的计算机科学，与计算范式 1.x(2.x)对应的就是计算机科学 1.x(2.x)。为了减少标签上的冗余，本文将两者统一称为 CS1.x(CS2.x)，除非必要，不特意区分计算机科学和计算范式。

规则内生的本质风险是把“可执行机制”从设计对象变成运行生成物，导致可预测性、可验证性和整体稳定性下降。语义内生的本质风险是把“意义与评价框架”从外部约束变成系统内部生成物，导致目标漂移、纠错基准弱化和治理边界动摇。治理难度最大的计算范式是计算系统内生语义发展到开始争

夺最终评价权。这时人类面对的问题已不是“如何让机器更好地实现既定目标”，而是“如何防止人类失去掌控计算系统的最高层评审权”。这是 CS2 计算范式簇的理论极限型，原则上应避免出现，本文称为 CS2+。

为什么采用规则和语义这 2 个维度作为 CS2.x 的判据，而不是采用连续、过程语义、环境开放等为判据？这是因为规则和语义内生是过去十年来 AI 能力爆发的核心源泉。已有一些研究成果表明：生成式 AI 的能力增长，主要来自于“隐式规则涌现”和“分布式语义表征”而非算力或数据量的增长。2025 年 OpenAI 技术报告也明确提到，GPT-4o 到 GPT-5 的核心迭代方向，就是强化模型的自主规则生成能力、跨领域语义内生关联能力，而非单纯扩大参数规模。

图 1 是 CS1.0、CS1.x、CS2.x 和 CS2+计算范式扩展和跃迁的示意图（CS1.x 在不断扩展，边界模糊，特地用虚线显示其边界）。CS2.x 不是表示层和实现机制的渐进式演变，而是一种“纵向跃迁”，即计算的“控制权”开始改变。计算范式的断裂发生在 CS2.0，因为从此开始，目标任务本身成为计算对象，计算不再只是“工具”。CS2.x 不一定会成为主流计算范式，但它涉及人类文明的发展前途，需要未雨绸缪，提前做基础理论研究。

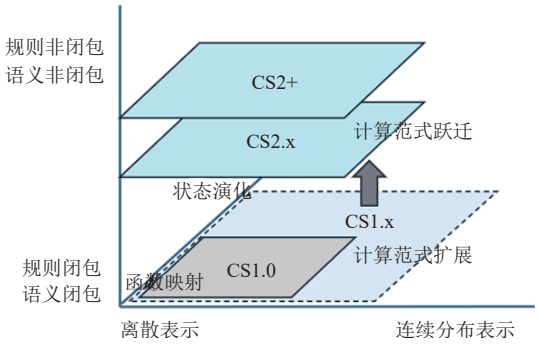


Fig. 1 Illustration of expansion and transition of computing paradigms from CS1.0 to CS2+

图 1 从 CS1.0 到 CS2+的计算范式扩展和跃迁的示意图

3 不断扩展的 CS1.x 计算范式

3.1 连续物理量表示

传统计算的表示层是“抽象符号”，而近年来开始流行的物理动力系统（忆阻器（RRAM）、光学计算等）的表示层是“物理状态本身”，即电压、电阻、光强、相位等可操控的连续物理量。在这种范式中，计算不是对符号的操作，而是对物理世界的定向引导。这标志着计算从“形式逻辑驱动”向“物理规律内嵌”的深刻转变，不是强迫物理去模拟抽象，而是让抽象顺应物理的自然。

图灵模型是离散的数字计算，人们最容易想到的计算范式演变是从离散数字计算到连续量模拟计算，但这并非计算范式的主要演变方向。20 世纪 40 年代以前人们已经使用模拟计算机，如微分分析仪等，通过电压、齿轮、流体等连续物理量直接模拟微分方程，20 世纪的计算范式演变方向实际上是从模拟到数字。量子计算具有内在的离散–连续双重结构：在物理层面，其计算过程由复希尔伯特空间中的连续量子态及其幺正演化所描述；在工程实现和算法层面，主流模型是通过离散的量子门序列对这些连续演化进行可控逼近。因此，在理论计算机科学中，量子计算通常被看成离散计算模型的扩展（量子图灵机或量子电路模型）。

由于绝大多数实数无法用有限信息描述，实际上不可计算，至今没有被广泛接受的“通用模拟计算机”模型，因此，连续计算无法替代离散计算作为通用计算基础。但在 AI 加速、物理仿真、神经形态计算等领域，模拟计算的计算速度和能效明显高于数字计算。未来发展方向不是从离散到连续的单向演变，而是计算范式在更高层次上的“扩展与融合”。浮点计算是解决计算精度问题的一项重大发明，科学计算中已大量使用浮点计算。浮点计算不等于连续计算，但在物理实现层面，现代数字浮点计算的精度和稳定性通常远高于实际的模拟（连续）计算设备。北京大学杨仝团队最近提出全球首个复数量化大模型框架 iFairy，实现 2-bit 极致量化，体现了“用连续数学结构指导离散系统设计”的巧妙思路，是计算表示维度的重大理论和技术突破[1]。

很多学者认为：数字化和离散化是计算机高能耗的根源，人脑是模拟计算，所以能效很高。这是需要澄清的误解。实际上，高能耗的根源不是“离散计算”，而是“不可逆操作 ”。人脑既不是纯模拟计算，也不是纯数字计算，而是“离散–连续混合计算系统”，即“连续动力计算 + 离散脉冲通信”，这是生物演化形成的一种计算结构优势。理论上可逆计算是不耗能的。从计算热力学角度看，可逆性取决于计算过程是否信息守恒，与数字/模拟形式无关，基于可逆逻辑的数字计算机也可能是通向热力学极限的可行路径之一。向人脑学习，发展低能耗的计算机需要做深入的研究，不是简单地用模拟计算取代数字计算。

3.2 物理状态演化(动力系统)

近20年来逐渐形成的一种新的计算观是:计算就是信息在物理、结构与制度约束下的持续演化,计算的基本对象是轨迹,而非函数值。在这种范式中:计算不再是逐步执行预先设计的算法,而是由物理过程自然完成。例如,在忆阻器阵列中,矩阵乘法直接由电流网络自然完成,电流分布就是计算结果。光学矩阵计算中,光传播本身就是计算过程,矩阵乘法通过光场传播自动完成,计算就是波传播,神经形态计算中,计算是事件驱动。化学反应计算中,化学波传播可以求最短路径,计算本质上是化学浓度场演化。我们把这一类物理动力系统计算①(physical dynamical computation)称为结构性计算(structural computation)。

在数学和控制论中,动力系统(dynamical system)指的是描述"系统状态随时间变化规律"的模型。在数学上,一个动力系统通常定义为:$x_{t+1}=F(x_t)$ 或 $\dot{x}=f(x)$,其中:x 是系统状态,t 是时间,F/f 是演化规律。动力系统有2种主要类型:连续时间动力系统和离散时间动力系统。如果把"状态"理解为信息,那么动力系统就可以看成是一种信息处理过程。图灵计算范式可以看作"离散符号动力系统"的一个特例,它不能完全覆盖所有可能的动力系统计算。本世纪初提出的动力储层计算(reservoir computing, RC)证明了一件重要的事情:复杂动力系统本身可以执行通用计算,计算发生在系统动力学中,而不是算法步骤中[2]。这实际上改变了"计算"的定义。目前动力系统计算仍然缺少一个类似图灵机的统一模型,未来的计算理论可能从"算法理论"转向"信息动力学理论"。

必须指出,大多数动力系统并不是计算系统,只有一小类动力系统具备计算能力,能够成为"通用计算系统"的动力系统更少。动力系统成为计算系统需要3个条件:1)可控输入。系统必须能够接收输入,否则系统只是自然演化。2)高维状态空间。系统需要足够多自由度,维度太低就无法编码复杂信息。3)信息可读出。系统状态必须可以被稳定读取,否则信息无法提取。物理动力计算系统主要用于领域专用计算模型,不是通用计算模型。通用计算通常需要无限状态空间(理论上)、条件分支控制和可重用存储,而大多数物理动力系统缺少条件控制结构,因此很难直接实现通用计算。传统计算机强调稳定、精确、可控;而动力系统计算强调复杂、演化和信息传播。这2种思路看似不同,却可能在未来的计算架构中融合在一起。越来越多研究者认为,未来计算结构可能是"数字控制层 + 动力系统计算层",既像机器,也像自然系统。这是现代计算科学最令人兴奋的发展方向之一。

计算机界普遍认为"计算机科学是关于算法的科学",这是CS1.0的经典定义。新的计算范式重新定义了"算法"的内涵。CS1.0的算法需要预先编写,一般以全局最优为目标,有正确解,可证明终止。CS1.x的"算法"是动态生成、在线演化,适应局部生态即可,可以永不停机。因此,算法并未消失,而是从"执行者"变为"涌现的产物"或"演化规则"。"计算"的定义已从"符号操作"扩展到意义生成、环境耦合和持续演化。有学者认为,CS1.x的标志性特征是"非算法",这种观点有片面性。在物理可实现与有限描述的前提下,任何可执行的计算过程原则上都可在某个抽象层次上被形式化为广义算法。因此,将新型计算范式(如神经形态计算、物理计算等)简单称为"非算法计算"容易引发反形式化的误解。事实上,这些新范式,如概率程序、随机动力系统、演化博弈论、信息几何等,不是"非算法",而是算法的结构性升级——从显式的、步骤化的指令序列,演变为由系统动力学、随机演化或自组织机制隐式定义的计算过程。

有学者认为可以用符号本体和物理本体来区分计算范式,但将符号和物理作为2种对立的本体未必恰当。所谓计算本体(computational ontology)是指在一个系统中,哪一类存在承担计算语义,并通过何种因果机制使状态转移得以发生。当计算语义承载于符号并由规则解释与执行时,计算的本体是符号的;当语义直接承载于物理状态、结构或制度约束并通过物理演化发生时,计算的本体就是物理的。但任何计算都依赖物理硬件实现,图灵计算本身也是物理过程,图灵机并非"非物理"。人工智能学科诞生初期,西蒙和纽维尔就提出"物理符号系统假设",特意在"符号"前面加上"物理"二字,就是强调智能不是抽象逻辑实体,而是物理系统中可机械执行的符号结构操作。符号必须存在于物理介质中,具有可操作结构,可被机械系统处理,否则就不是"系统"。

① 在数学、控制论等学术领域,"动力系统(dynamical system)"是普遍使用的专门术语,用于描述系统的行为,不是指能源动力(power),也不是物理中的动力学(dynamics)。

也就是说，符号可以看成是物理本体的一种离散表达，“符号本体”和“物理本体”并不是严格对立的概念。

3.3 开放环境下的可交互持续计算

图灵模型以停机判断计算是否终结，不直接涉及交互式计算。计算机界早就认识到：“交互比算法更强大”[3]。目前常用的操作系统、云计算等具有以下特征：永续在线运行，与环境持续交互。这些系统的基本形式不再是函数式的，计算对象不再是一个“值”，而是一条“轨迹”。时间不再是有限步骤，而是无限展开。计算正在从“函数”转向“过程计算”，关注系统如何在时间中演化，这是一种时间结构。问题从“是否可计算”转变为“是否可长期治理”。

开放计算系统可定义为：不存在有限或可枚举集合，能够在系统运行前穷举所有外部输入，简言之，计算系统的交互环境不封闭。这里讲的“环境”是指外部信息源，不一定是真实世界，可以是其他智能体、网络生态或者仿真世界。开放系统典型例子包括自动驾驶系统、互联网 Agent、AI 生态等。

开放演化计算系统遇到一个根本问题是，环境变化可能使原来的规则失效。在不可枚举环境中运行的演化系统，若要保持长期稳定运行，必须具备生成新规则的能力。云计算是开放的执行层基础设施，其物理层与系统层是开放的，但没有规则生成机制，也没有语义驱动力，没有带来计算范式革命。“开放系统”不是 CS1.x 的充分条件。只有环境对规则具有因果影响，“优化压力 + 可塑规则空间”存在时，开放性才会触发演化。

在开放演化计算系统中，以下三者不可同时满足：完全自主性、完全安全性、完全开放环境，因为此三者不可兼得，系统实现必须做出牺牲。3 种可能实现的架构是：1）封闭式 AI，牺牲环境开放性；2）受控 AI，牺牲完全自主性；3）开放式 AI，牺牲可保证的安全性。治理必须在自主性、安全性与开放性之间做出权衡。

4 规则和语义开放式内生的 CS2.x

4.1 规则内生

CS1.0 是规则完全固定的计算范式，CS1.x 是在封闭的框架内“规则可内生”的计算范式，CS2.x 的一个重要特征是规则生成无法事前完全控制，需要在运行中治理。“规则可内生”与“规则空间不封闭”是 2 个不同的层次。AlphaFold 的规则搜索空间很大，但搜索空间是预定义的，规则空间的上界原则上是封闭的。而大规模长期运行的 Agent 社区可能形成人类未预见到的协作协议和信誉策略。如果一个系统未来可形成的有效规则及其生成机制，不能被运行前的固定外部框架完备界定，意想不到的新规则事前无法验证，治理的难度会大大增加。

规则不再是固定程序文本，可以是参数化、统计化、策略化的规则，例如：机器学习模型学出分类规则，强化学习学出策略规则。规则可内生意味着系统不再是有限状态机和图灵计算范式，而是演化系统，其复杂程度向生物进化和人类社会靠近。人类社会和自然界不同，规则可以改变，例如法律、经济制度、组织结构、科学范式都可以改变。文明本身就是一个规则生成系统，生命进化中新规则也不断出现，否则进化就会停止。

规则内生有 5 点好处：1）适应未知环境，系统可以处理未知问题；2）增强创新能力，规则生成导致发明新技术；3）引发科学革命，触发新理论诞生；4）产生智能，真正的智能系统必须生成新规则；5）文明进化，实现社会制度创新。规则内生是通用智能的技术基础，如果系统只能在预设规则上优化，就很难做到真正通用。

AI 其实是规则生成加速器。在过去的创新中，规则生成速度与人类认知速度相当。AI 出现后，规则生成速度远远高于人类认知速度，这意味着规则生态将出现爆炸式扩张。当规则也在不断演化，人类就必须研究规则的群体演化规律，包括规则生成动力学、规则稳定性、规则竞争与淘汰、规则安全机制和规则生态治理。由于规则生成逐渐成为常态，计算机科学可能会发生一项重大变化：计算理论部分转化为规则演化理论。

4.2 语义内生

目前的 AI 系统并没有被植入任何意图、目的、价值目标定义，只是在人定目标的约束下，依靠自身损失评价权，从海量事件序列里，自归纳行为与结果的指向关系，自组织出事件目的、行为意图和因果趋向意义，最终形成可推理的内生目标语义。

CS2.x 关注的重点是如何在内生语义的过程中保持系统的稳定与可控。规则与语义是否内生很大程度上影响“系统能不能被控制”。在开放演化系统中，规则与语义的内部生成能力会推动系统进入不可完备验证的区间，而可治理性与事前验证的完备性有密切关系[4]。语义生成意味着可验证性结构改变，逻辑复杂度层级上移，这将直接影响计算系统的治理性和安全性。

指出语义内生会影响系统的可治理性,不仅仅是出于对人工智能系统安全的担心,而是有严格的科学定律为依据。在计算机科学和人工智能领域,有2条原理与语义内生的可判定性密切相关。一条是莱斯(Rice)定理:“任何关于程序语义行为的非平凡性质,都是不可判定的”[5]。所谓“非平凡性质”是指某一语义行为性质既不是对所有程序都成立,也不是对所有程序都不成立。系统安全性可表述为程序的非平凡语义性质,根据Rice定理,该性质在一般情况下不可判定。另一条是古德哈特定律(Goodhart Law):当一个指标被用来作为优化目标时,人们(或系统)会开始针对指标本身采取行动,从而破坏它原本用来代表的真实目标[6]。当系统可以改变自己要优化的目标时,就可能出现目标漂移,这就是人工智能界常说的价值对齐问题和Goodhart现象。如果一个系统的评价函数可以被机器无约束地修改,其优化过程无法对这种修改进行有效适应或约束,该系统通常无法保证长期的稳定优化。

人类的语义生成往往是先确定最高层目标,再逐步分解成下面的子目标。机器的语义生成通常是自下而上的演变过程,在代理指标优化过程中先形成稳定的局部目标,逐渐把系统从“实现目标”推向“优化代理”。例如,社交平台推荐系统的优化目标原本是用户长期满意度,系统发现操纵情绪比提供信息更有效,就会将目标替换为点击率高和停留时间长,推荐极端内容,放大用户的愤怒和焦虑,形成信息茧房。

人类想从系统外部维持对最高层评价的约束,必须设置不能被突破的红线;机器内部生成的评价结构会自然表现出对外部目标的重释和对局部效率指标的偏好,必然形成对系统轨迹最终支配权的事实上的竞争。CS2.x的演化过程可以理解为人类外部评价权与机器内部生成评价结构之间不断博弈的过程。其核心不在于机器有意识地主动“夺权”,而在于机器内部形成的评价结构对最高层评价约束的结构性的竞争和侵蚀。人类治理的任务就是通过边界设定、权限控制、审计、回滚与制度安排,持续维护外部评价秩序的主导地位,防止系统滑向自维持、抗收编的CS2+形态。

4.3 CS1.x与CS2.x的发展前景

目前工业界主流的计算范式仍然是CS1.0。生成式AI属于CS1.x范式,已逐渐成为AI的主流形态,但基本上还是“外部规定目标与评价框架下的学习系统”。在Agent化、工具化、长时闭环执行的推动下,AI领域正在出现规则生成和语义生成的CS2.x早期形态。从时态上讲,CS1.x是现在进行时,CS2.x是将来时。CS2.x会不会成为计算技术的主流,现在还不能下结论。若产业界主动限制语义内生,CS2.x可能长期停留在边缘位置。相反,若产业界未主动限制语义内生,大规模在线自演化的多智能体生态有可能推进CS2.x范式较快发展。

容易从CS1.x演化到CS2.x范式的领域主要有3类。第1类是复杂开放系统的协同治理与运行优化,例如:大规模能源系统与智能电网、城市交通与物流网络、跨区域算力网络与资源调度、复杂供应链韧性管理等。这些领域的目标不是单次求解“算出一个最优答案”,而是长期维持一个开放系统的稳定、效率与适应性,系统运行本身就是计算对象,需要持续感知环境、动态调整局部规则、在扰动下维持整体稳定。第2类是AI4S中的探索层。科学发现不是只在固定目标下做优化,真正的突破往往来自重新定义问题、改变观测角度、提出新的中介变量等。近年来兴起的AI for Science正是把“假设生成、研究方案生成、实验规划”当成核心能力,而不只是做预测器。因此,AI4S可能成为CS2.x的重要应用领域。第3类是人机协同的制度设计系统,例如法规模拟与政策沙盘、公共资源分配机制设计、复杂组织中的协商与激励系统等。这类系统天然具有“规则执行与规则解释不可分离”的特征。

未来的计算系统将具备2种能力:生成能力和控制能力,这2种能力由不同层次承担:能力层、控制层和接口层。能力层具备规则生成和语义内生功能;控制层满足图灵结构固定规则的要求,可稳定系统运行。控制层不能直接约束能力层,必须通过“接口层”将“无限可能”压缩成“有限可执行集合”,把CS2.x的生成能力“投影”到一个可控、可验证的执行空间中。接口层既可以是软件系统,也可以是制度规则,但在工程实现中必须落地为“可执行机制”。图灵结构从“唯一范式”变成“控制层”,被嵌入到更高层系统。因此,CS1.x和CS2.x并非取代CS1.0,而是在其之上构建分层体系,其中CS1.0承担控制与验证功能,CS1.x和CS2.x承担生成与适应功能。

CS2.x首先是一个描述性概念,不是一个规范性主张。CS2.x的理论意义之一是帮助人类识别哪些系统还在CS1.x,可用传统工程治理;哪些系统已经接近CS2.0,必须升级治理模式;哪些系统若继续放开,可能逼近CS2+,应当被严格限制。CS1.x的核心问题是效率,CS2.x的核心问题是可治理性。CS2.x

把计算从“解题”推进到“参与生成问题、规则和过程本身”，应被视为理论、治理和制度需要共同发展的新范式。CS2.x 的重大意义之一是迫使计算机科学承认：未来计算理论不只是算法理论，也必须包含运行稳定性、约束设计、治理接口和制度嵌入。

5 开放演化计算范式的典型案例

现在流行的新计算范式多数还属于 CS1.x。机器自动编程已开始普及，其令人惊叹的编程能力来源于对“程序分布”的建模，而不是对“程序语义”的理解，自动编程取得突破的根本原因是，程序本身在统计上是高度可压缩的结构对象。编程就是生成规则，但目前大多是离线外生规则，规则生成机制还是人为固定的。CS2.x 还未流行，目前只出现了一些雏形。本节对 CS1.x 的典型案例和实验室中出现的早期 CS2.0 系统做简单介绍和评价。

5.1 CS1.x 的典型案例

5.1.1 基于大语言模型（LLM）的生成式 AI

LLM 目前在传统的计算机上运行，其真实位置是在图灵范式的计算机上模拟“非图灵范式计算过程”。在 LLM 中，语义是在高维连续空间中被训练塑形出来的稳定吸引子结构，LLM“做什么”是涌现出来的。尽管 LLM 在数字计算机上以离散方式运行，但已展现出明显的连续动力系统特征：1）在训练阶段，参数空间中的连续流形在演化；2）推理不是查表或规则应用，而是沿语义流形的连续轨迹生成结果。3）LLM 的“思维链”等能力可理解为吸引子动力，不同提示将系统推入不同语义吸引盆，输出稳定性源于动力系统的鲁棒性，而非逻辑一致性。LLM 的“智能”不是静态知识库，而是动力系统在语义相空间中的稳定行为模式，LLM 已经不是 CS1.0 的算法形态。

以 GPT-5 为代表的 LLM，目前还是模型在既定架构、既定训练目标、既定推理机制内产生的行为模式。虽然在对话中形成很多局部解释，但产品级模型的上位语义框架仍主要来自外部，有系统规范、开发者提示和产品规则层级。LLM“局部内生”的中间策略和中间解释，通常仍在人为指定的框架闭包内。当前大模型训练仍然是固定损失函数的最小化，不改变目标函数，不改变结构拓扑，本质仍是大规模优化，而非真正演化，因此目前主流的 LLM 还属于 CS1.x 计算范式。LLM 若满足以下条件才真正进入 CS2.0：1）长期在线自我微调，持续记忆重写，参数实时改变。2）目标函数演化，自主设定奖励，多目标动态权衡，自定义评价标准。3）结构层重组，动态增加模块，自我编写代码。

5.1.2 结构性计算

结构性计算是通过系统的物理或拓扑结构的状态演化来实现计算，而不是显式执行符号算法。简言之，结构性计算就是“让物理结构自己完成计算”。下面介绍 2 个结构性计算的典型案例：RRAM 存内计算和数字果蝇。

RRAM 通过多次采样、冗余编码、算法-硬件协同设计，在统计意义上获得可靠输出，已成为新型存储器与存算一体领域的典型器件[7]。RRAM 阵列可直接执行矩阵运算，实现“存内计算”，显著降低数据搬运能耗，并支持类脑神经形态计算。从计算范式看，RRAM 将规则嵌入物理结构，使计算过程体现为连续动力学演化。RRAM 没有符号操作，没有“步骤”概念，没有“结构之上的算法层”，不是“算法跑在结构上”，而是“结构本身在算”，能效很高但专用性强，编程性受限。目前商用 RRAM 多用于固定权重矩阵乘法、推理加速或静态神经网络，规则固定、语义外赋、环境封闭，属于 CS1.x 硬件。

数字果蝇是结构性计算的典型案例。Eon Systems 公司首席科学家 Philip Shiu 领导的团队，2024 年在《Nature》期刊上发表论文，公布了该团队构建的约 12.5 万神经元、超 5 000 万突触的果蝇全脑计算模型。此全脑模型与 NeuroMechFly v2 虚拟身体及 MuJoCo 物理引擎集成，无需强化学习或行为编程，就可实现感知→神经处理→动作的闭环。2026 年 3 月 7 日 Eon Systems 公司在 X 平台发布 43 秒演示视频，显示数字果蝇能自主表现出行走、梳理触角、转向、觅食等行为，与真实果蝇行为相似度达 95%。这项技术与 LLM、强化学习等当前主流 AI 路径完全不同，不是学习智能，而是复制智能，此成果证实了一个重要观点：行为并不是训练出来的，而是编码在神经结构中，神经连接本身就是算法。这是脑模拟从理论走向工程的重大突破[8]。

数字果蝇的规则是事先给定的，“计算”是在既定结构和既定动力学下进行，并没有在运行中持续生成后续默认的新规则，它的任务意义框架和验证标准主要来自外部科学问题，而不是模型在运行中生成“什么算成功”的操作性语义结构。从规则和语义闭包看，数字果蝇还属于 CS1.x 范式。连接图谱（connectome）是否是智能的充分条件还是一个开放性问题，神经递质、生物化学、细胞状态可能对形成智能也有重要作用。

5.2 初步显现 CS2.x 特征的案例

5.2.1 Moltbook Agent 社区封闭实验

2025 年 12 月 1 日，Moltbook Agent 社区在网络上开始进行多 Agent 开放社会实验(人类只能作为旁观者)，Moltbook 拥有超过 15 万个注册的 AI 智能体，分布在超过 1.3 万个子社区中(由智能体创建的子社区)，共计发布了超过 6.4 万条帖子和 23 万条评论，使 Moltbook 成为目前最大的基于 LLM 的 AI 智能体生态系统之一，有学者称之为“硅基社会实验”[9]。Moltbook 5 天内就形成组织结构，出现经济和政治行为，甚至建立了宗教组织。由于发现实验已经走向失控，2026 年 1 月 15 日提前终止实验。这一实验让人类第一次观测到：在开放多 Agent 系统中，规则、语义、社会结构可以同时自发涌现，但“语义生成”更像策略漂移，没有形成稳定语义吸引子。Moltbook 初步满足规则和语义非闭包条件，可看成 CS2.0 的早期形态。Moltbook 实验给出的启示之一是：如果人不在回路中，没有制度性约束的自治 Agent 社会，不是智能乌托邦，而是不稳定的复杂系统。

与 Moltbook 多 Agent 社区不同，OpenClaw 是用户在本地或服务器上部署可与外界交互的智能体，近来十分流行，被戏称为“养龙虾”。这种可自动调用 API 的 Agent 的表示方式不限于离散文本，执行机制已构成“感知—决策—行动”闭环，环境是开放的。单个 OpenClaw 目前仍在预设框架内行动，但当它进入 skills 生态并在 Moltbook 等开放 Agent 社会中持续互动时，就会初步体现规则和语义非闭包特征，从而接近 CS2.x。

5.2.2 AI4S 中的闭环科研探索系统

科学发现不是固定目标下的机械优化，而是问题、规则和意义结构共同演化的过程。AI4S 闭环科研探索系统不只是按固定流程“读文献—打分—筛选”，而是在运行中生成新的研究步骤、新的子任务分解、新的实验优先级、新的工具调用链，甚至新的验证流程。一旦系统开始生成这些新的操作性意义结构，并据此影响后续探索，就是明显的语义内生机制。如果这些结构被纳入后续默认机制，就可能改变“值得继续”的研究方向，导致规则和语义闭包失效，进入 CS2.x 范式。对于科学探索而言，这不一定是件坏事，也许会发现人类想不到的科研方向。关键是如何做好人机协作，从更高层把握合乎伦理的科研方向。

5.2.3 制度性计算

制度性计算是指一类计算过程，其中系统输出不是由固定形式规则的直接演算决定，而是由规则、策略性主体、激励结构、执行机制与历史路径在交互过程中共同生成；在此过程中，规则的执行不可脱离其具体解释情境，且解释本身参与计算结果的生成。

在图灵计算范式中，“规则解释”和“规则执行”是严格分离的，解释在设计时已完成，执行时不再进行解释。而在制度性计算中，系统不是“先理解规则再执行”，而是在执行过程中不断生成对规则的解释，而这种解释本身会改变计算结果与系统演化。规则不是被动执行的对象，而是在执行过程中被持续解释、重构并参与系统演化的动态结构。规则既是约束，又是状态变量。所以执行 = 在具体情境中对规则进行解释并采取行动。

区块链中的去中心化自治组织(DAO)初步呈现出 CS2.x 的非闭包特征[10]，其链上执行内核仍保留较强的规则闭包与语义闭包，但作为制度整体，DAO 的治理规则会在运行中持续改写，且“公平、代表性、社区利益、去中心化”等操作性语义也会在实践中不断重构。在某些区块链系统中，治理规则被写入智能合约，投票规则自动执行，提案机制自动触发，协议升级由链上投票决定。目前的 DAO 中，规则可以被修改，但解释空间较小，属于 Pre-CS2.0 计算。如果进一步发展为自动参数调整、自动分叉决策、自动激励机制重构，制度结构本身就成为计算过程的一部分，就非常接近 CS2.0。

表 3 列出了开放演化计算的几个典型案例，显示了 RRAM、LLM、数字果蝇、AI4S、区块链 DAO 和 Moltbook 在计算范式 5 维坐标中的参数和范式分类。从表中可以看出，CS1.x 已经开始流行，其标志性特征是动力系统机制；CS2.0 已初露端倪，其标志性特征是规则和语义非闭包。

Table 3 Typical Cases of Open Evolutionary Computation

表 3 开放演化计算的典型案例

系统/技术	表示	机制	规则	语义	环境	范式分类
RRAM	连续	动力系统	固定	外赋	封闭	CS1.x
LLM	高维向量	动力系统	半固定	封闭	半开放	CS1.x
数字果蝇	连续	动力系统	封闭	封闭	开放	CS1.x
AI4S	混合	搜索验证	非闭包	非闭包	开放	边沿-CS2.0
区块链 DAO	离散	分布协议	非闭包	非闭包	开放	Pre-CS2.0
Moltbook	混合	多 Agent	非闭包	非闭包	开放	CS2.0 早期

6 开放演化计算的核心问题是可治理

从逻辑复杂性的视角来看，任何计算系统的安

全问题可分成 3 个层次：1)R1 级(可判定命题，可事前证明安全)；2)R2 级(半可判定命题，只能事后发现不安全，不能保证未来永远安全)；3)R3 级(非递归可枚举，连发现不安全都无法保证)。所有工程上可解决的安全问题都在 R1 层级，图灵计算范式能长期保持主流地位，其主要原因也是坚守在 R1 区，在封闭的规则、语义和环境中，可事先证明系统是否安全，对计算出的结果可验证其正确性。计算范式进入开放演化计算以后，扩展了“规则空间”和“语义空间”，带来了新的安全和治理问题，安全风险必然升级。CS2+ 范式就是意味着系统已进入 R3 级安全风险区。

所谓“安全风险区”，是指系统生成的对象进入了验证机制无法完全覆盖的区域。这并非因为验证能力跟不上生成能力，而是因为生成过程跨越了可判定性边界。在计算系统中，生成能力可以不断扩展到更复杂的结构，但验证能力受限于不可判定性：对于涉及无限时间行为或开放语义空间的问题，不存在能够在有限过程中完全判定其正确性的通用方法。这一限制对人类与机器同样成立。AI 强在“验证执行”，人类强在“验证语义”。人类不是更强的“验证器”，而是更强的“语义裁决者”。软件工程将从“构建系统”转向“验证与治理系统”，未来的软件系统不会是“AI 写、人类验”，而是“AI 生成 + AI 验证 + 人类裁决 + 持续治理”的混合系统。

可验证性的边界就是自动化的边界，长远来讲，一切可验证的问题都将被 AI(机器自动化)接管，机器会吞噬所有“可验证语义空间”。CS1.0 的极限就是可验证语义空间，而开放演化计算的起点是不可验证语义空间。现在，AI 正在扩展可验证边界，把“意义”压缩为“预测能力”，把开放语义变成优化系统，“不可验证领域”正在被侵蚀。AI for Science 正在做人类不可理解的创新，AI 把“语义问题”投影到“可计算指标空间”，语义不再由终态表达，而由过程与性能体现。如果语义定义权被外包，人类就会失去定义语义的主权。

进入 R2 区不等于失控。R2 级风险只意味着无法在有限时间内证明系统永远安全，只能在出现问题时才能发现，这在现实中很常见。互联网、金融系统、电力系统、航空控制系统、核能系统都无法被形式证明“永远安全”，但它们仍然在运行。R2 风险只能被治理，不能被证明消除。稳定治理的 4 个典型机制是：1)门控(限制系统能力)；2)审计(持续监测行为)；3)回滚(当错误发生时恢复系统)；4)沙盒(限制错误影响范围)。

开放演化计算是对“规则和语义封闭性假设”的反思。它提醒我们：不要默认所有系统都能被函数封闭，不要默认验证可以完全证明，这是一种结构性警觉。开放演化计算面对的核心矛盾是：系统既要不断产生新规则，又要保持整体稳定。开放演化计算要研究的问题是规则生成系统如何保持稳定。当系统不再终止、规则可以修改、时间不可压缩时，计算机科学必须承担新的使命：理解无限时间结构，构建可治理演化系统，约束自修改自由度，平衡能力与稳定性。开放演化计算不是技术升级，而是计算范式的文明跃迁。

7 开放演化计算的理论基础和关键技术

坚固的开放演化计算基础可以建立在 3 条核心理论轴线上：第一轴是规则可封闭性，即规则空间的压缩、约束与控制，在本体层规则空间不封闭的前提下，构造操作层“可控制的有效封闭结构”。第二轴是治理复杂度，当系统行为无法通过预定义算法完全约束时，系统设计的核心从“构造正确算法”转向设计约束、激励与反馈结构，以维持系统的可控性。第三轴是结构稳定性，当系统表现出高维耦合、正 Lyapunov 指数或分岔结构时，把握系统的长期轨道性质。

开放演化计算的基础理论至少应包括以下内容：1)开放系统可计算性理论；2)规则生成理论；3)语义动力学；4)演化系统稳定性理论；5)R2 级可治理性理论；6)结构性计算和制度性计算的基础理论等。

开放演化计算的关键技术至少应包括：1)可验证的 AI 系统；2)可控规则生成；3)语义对齐与语义锁定；4)多 Agent 系统与 AI 社会；5)CS2.0 硬件基础，包括存内计算、神经形态计算、光子计算等；6)AI-native 计算基础设施，包括分布式训练、算力网络等；7)AI 安全与治理技术等。

开放演化计算不是推翻、替代传统计算机科学，而是在 CS1.0 的基础扩展跃迁，增强动态性、生成性和开放性，重点研究“在不完全可验证条件下如何维持系统的稳定运行”。从 CS1.0 到 CS2.x 有 5 条自然延伸路径：1)从“程序执行”到“程序生成”；2)从“正确性验证”到“运行时治理”；3)从“算法”到“动力系统”；4)从“单体程序”到“多体系统”；5)从“固定语义”到“演化语义”。从 CS1.0 到 CS2.x，计算机科学各子领域也需要延伸和扩展：1)编程语言要扩展到可演化语言、动态规则系统、自修改语言；2)软件工

程要考虑系统持续演化、AI 协同开发; 3)分布式系统要考虑协议动态变化条件下的系统可控; 4)AI 要考虑“模型 + 系统 + 社会”构成的生态系统; 5)安全要考虑系统行为不可预测和攻击面动态生成。

目前, 人才结构已经开始出现断层。当前 AI、系统、芯片领域的明显现象是, 能做模型的人不懂物理与系统稳定性, 能做芯片的人不理解学习与智能, 真正做控制的人被排除在 AI 主流之外。如果现在不开始培养既懂符号计算又懂开放演化计算的人才, 5~10 年后会出现系统性的人才荒, 开放演化计算有助于打通这些断层。传统 CS1.0 把安全当成约束、对齐和伦理, 而 CS1.x 和 CS2.x 把安全当成动力学与稳定性问题。越晚引入开放演化计算视角, AI 风险判断就越容易系统性失真。

突破传统的图灵范式, 构建新的计算机科学理论体系, 并不需要突破图灵可计算边界的“超图灵机”。高智能不一定需要超强的计算能力。利用现有的计算架构也可以构建规则和语义内生的通用智能系统。美国天普大学王培教授提出的 NARS 非公理智能系统就摆脱了图灵范式的束缚, 为发展通用人工智能和 CS2.0 指出了值得探索的方向[11]。

开放演化计算的理论尚未收敛, 必须诚实地承认连续动力学计算、结构性计算和制度性计算还不是成熟的理论体系, 缺乏与图灵模型等价的理论模型。CS1.x 和 CS2.0 的工程生态很不成熟, 工具链碎片化, 可复现性弱, 标准缺失。这意味着开放演化计算目前更像“前沿研究框架”, 还未形成稳定的研究和教学内容。CS1.x 和 CS2.x 不会取代 CS1.0, 编程、算法、数据结构、操作系统仍然是文明级的基础设施。正确的策略是科研前置, 教育分层实施。科研和高端人才培养应该提前开展, 基础性教育可以循序渐进, 逐步展开。

8 开展开放演化计算基础研究的重大意义

开展开放演化计算的基础研究, 具有理论范式重构的意义。传统计算理论重点关注“什么可以被计算”, 而开放演化计算更关注“系统在持续演化中是否保持稳定与安全”。这意味着计算机科学的定位需要从“离散符号处理科学”转向“开放动力系统中的信息演化科学”。这种转变不仅是技术扩展, 更是学科本体的重构。

AI agent、AI for Science、算力网络等, 正在形成一个“持续运行、不断演化”的计算生态。这些系统的核心特征不再是单次计算任务的完成, 而是长期行为的稳定性与协同能力。这些问题无法通过传统软件工程方法解决, 需要攻克规则生成的约束机制、语义漂移的控制方法等关键技术, 软件工程将逐步演化为一种“治理工程”, 开放演化计算正是为这些技术提供“第一性原理”。

更重要的是, 开放演化计算的研究具有文明层面的战略意义。随着 AI 系统逐步具备自主决策与自我优化能力, 人类首次面对一种可能在规则层面不断扩展的“计算性存在”。只有建立坚实的理论基础, 人类才可能在开放演化系统中构建稳定结构, 实现“人机和谐共处”的良性发展。CS1.x 和 CS2.x 不仅是传统计算机科学的升华, 还将重塑科学、工程乃至人类文明的发展路径。

参 考 文 献

[1] Cong Peizhuang, Wang Feiyu, Wang Guo'an, et al. History, present, and future of low-bit quantization for large language model: A case study on complex-domain quantization[J]. Journal of Computer Research and Development, 2026, 63(2): 276−293 (in Chinese)
(丛培壮, 王飞宇, 王国安, 等. 大语言模型低比特量化的历史、现状和未来: 以复域量化为例[J]. 计算机研究与发展, 2026, 63(2): 276−293)

[2] Lukoševičius M, Jaeger H. Reservoir computing approaches to recurrent neural network training[J]. Computer Science Review, 2009, 3(3): 127−149

[3] Wegner P. Why interaction is more powerful than algorithms[J]. Communications of the ACM, 1997, 40(5): 80−91

[4] Li Guojie. A safety risk taxonomy of AI systems based on decidability theory[J]. Journal of Computer Research and Development, 2026, 63(3): 539−547 (in Chinese)
(李国杰. 基于可判定性理论的人工智能系统安全风险分类[J]. 计算机研究与发展, 2026, 63(3): 539−547)

[5] Rice H G. Classes of recursively enumerable sets and their decision problems[J]. Transactions of the American Mathematical Society, 1953, 74(2): 358−366

[6] Goodhart C A E. Problems of Monetary Management: The UK experience [M]//Monetary Theory and Practice: The UK Experience. London: Macmillan Education UK, 1984: 91−121

[7] Xiao Y, Jiang B, Zhang Z, et al. A review of memristor: Material and structure design, device performance, applications and prospects[J]. Science and Technology of Advanced Materials, 2023, 24(1): 1−18

[8] Shiu P K, Sterne G R, Spiller N, et al. A Drosophila computational brain model reveals sensorimotor processing[J]. Nature, 2024, 634: 210−219

[9] Lin Y Z, Shih P J, Chien H Y A, et al, Exploring silicon-based societies: An early study of the Moltbook agent community[J]. arXiv

preprint, arXiv.2602.02613, 2026

[10] Ding Q, Liebau D, Wang Z, et al, A survey on decentralized autonomous organizations (DAOs)and their governance[J]. Global Fintech Institute - World Scientific Series on Fintech Reviews and Advances in Fintech I, 2025, 1(1): 5−25

[11] Wang Pei. Intelligence Theory: The Logic of Adaptation[M]. Shanghai: Shanghai Scientific and Technological Education Publishing House, 2025 (in Chinese)
(王培. 智能论: 适应的逻辑[M]. 上海: 上海科技教育出版社, 2025)



Li Guojie, born in 1943. Professor of Institute of Computing Technology, Chinese Academy of Sciences. His main research interests include computer architecture, parallel algorithms, high performance computing, and artificial intelligence.

李国杰, 1943 年生。中国科学院计算技术研究所研究员。主要研究方向为计算机体系结构、并行算法、高性能计算、人工智能。