arXiv:2603.10720v1 [cs.DS] 11 Mar 2026

具有联合移动的可编程物质次线性时间重构

ManishKumar 印度理工学院鲁帕尔分校，印度**Othon Michail** 利物浦大学，英国**Andreas Padalkin** 帕德博恩大学，德国**Christian Scheideler** 帕德博恩大学，德国

摘要

我们研究几何变形虫机器人结构的集中式重构问题。一组 n 变形虫机器人占据三角网格上的节点，并可通过扩展与收缩操作进行重构。我们重点研究联合移动扩展：变形虫机器人可以并行扩展和收缩，从而实现较大子结构的协调运动。先前的工作提出了这一扩展，并在元模块等附加假设下分析了重构问题。

与此不同，我们通过将研究范围限定于集中式算法，探究不依赖这些假设的重构内在动态，并将分布式解决方案留待未来研究。我们研究两类变形虫机器人结构 A 和 B之间的重配置问题：对于每个结构 $S \in A$，目标是计算一个调度，将 S 重构为某个结构 $S' \in B$。我们的重点是次线性时间算法。

我们通过证明任意结构都能在 $O(\sqrt{n}\log n)$ 轮内重构为规范的线段结构，肯定地回答了Padalkin等人（Auton. Robots，2025）提出的开放问题：模型内是否可能存在一个亚线性时间的通用重配置算法。此外，我们还给出了一种常数时间算法，可将任意螺旋结构重构为一条线段。这些结果得益于新的常数时间原语，它们能够促进高效的并行移动。

我们的研究结果表明，联合移动模型无需辅助假设即可支持亚线性时间重构。一个核心的开放问题是，能否在该模型内以多对数时间甚至常数时间实现通用重配置。

2012年ACM主题分类计算方法 → 合作与协调；计算理论 → 计算几何

关键词与短语amoebot模型、可编程物质、模块化机器人系统、重构

经费*Andreas Padalkin* 和*Christian Scheideler*：这些作者获得了德国研究基金会项目 SCHE 1592/10-1 的资助。

致谢 感谢 Christian Bauer 和 Leo Decking 实现了本文介绍的算法。

1 引言

可编程物质由许多微小且相同的机器人模块构成，它们可以按照可编程的方式改变物质的性质，例如形状、颜色或密度 [18]。本文从集中式视角研究几何 Amoebot 模型 [7, 8, 9]的重构动态。在该模型的分布式版本中，一组 n 机器人（称为变形虫机器人）被放置在三角网格上。每个变形虫机器人都可以与相邻变形虫通信，并通过扩展到相邻的空节点中，然后

再收缩进入该节点。由于移动和通信都是在节点之间局部进行的，该模型中的许多问题都有一个自然的下界：$\Omega(D)$ 轮，其中 D 是结构的直径。

为克服这一局限，先前的工作提出了联合移动扩展 [14]。这一扩展允许多个变形虫机器人以协调的方式共同移动。这一扩展允许多个变形虫机器人以协调的方式共同移动。例如，变形虫机器人可以在扩张或收缩时推拉其他变形虫机器人，从而使结构的大部分能够并行移动。这引出了一个重要问题：我们能否利用联合移动，更快地重构变形虫机器人结构？这里所说的“更快”，是指以次线性并行时间（以离散轮次计）运行，或者在可能的情况下，甚至在多对数或常数时间内完成。本文关注的是这种方法原则上是否可行，因此自然将重点放在该模型的集中式重构动力学上。

能够全局影响结构的驱动操作有时称为线性强度操作，这体现了它们具有足够的力量，可以移动规模至多为线性的集合。尽管在某些系统中，这类操作可能是一个很强的假设，但它们已被纳入多种理论模型 [2, 3, 4, 5, 13, 19]，并且在一些现实场景中是合理的，例如低黏度环境中的微纳机器人系统、DNA纳米技术，以及可编程物质；在这些场景中，单个局部操作就能直接导致整个系统发生重构。它们也为一种动力学提供了简单的理论抽象；在实践中，这种动力学可以通过联合移动来实现，由多个机器人共同承担线性强度操作的负载。本文通篇使用移动 和移动 指代驱动操作。

Aloupis 等人 [4, 5] 研究了一种模块化机器人系统模型，称为晶体机器人 [17]。晶体模型的二维版本将模块表示为二维网格上的正方形，形成由附着在相邻模块上的模块组成的连通形状。每个独立模块都可以进行扩展和收缩：将其一个面向外延伸一个单位，之后再将其缩回。由于模块彼此附着，一个模块的扩展或收缩可能使规模达到线性的组件发生移动。在 [5], 中，他们针对晶体模型提出了一种通用的集中式重构算法，该算法能够将任意一对模块数量相同的连通形状 S_I， S_F 转换 S_I 为 S_F，所需轮次为 $O(\log n)$ 轮。需要注意的是，在晶体模型中，某些类别的形状若不使用元模块（或其他假设）则无法进行重构，而这些算法本质上依赖于元模块的使用。

近期研究表明，在使用元模块等额外假设的情况下，几何 Amoebot 模型中的联合移动同样可以非常强大 [14]。然而，在不依赖额外假设的情况下，该模型本身的能力仍不明确。在本文中，我们研究带有联合移动的几何 Amoebot 模型中的集中式重构问题，不依赖此类额外假设，从而聚焦于模型动力学本身的局限性。

我们考虑集中式算法，其中全局调度器了解完整结构，并决定每一轮中所有变形虫机器人的移动。这使我们能够聚焦于核心问题：原则上重构能有多快？给定两类阿米巴机器人结构，目标是设计移动序列，将一类中的每个结构重构为另一类中的某个结构。在此语境下，算法是对源类中所有结构的这些移动序列的形式化描述。

我们的主要成果是首个模型内通用次线性时间重构算法。我们证明，任意连通的变形虫机器人结构都可以重构为一条直

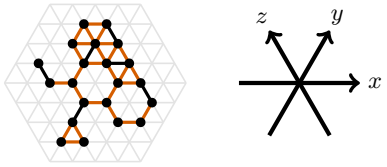


图 1 Amoebot 模型与坐标轴。黑色节点和边表示变形虫机器人占据的节点和边。红色边表示变形虫机器人之间的键。灰色边表示三角网格 G_Δ。其他所有图中均省略网格。

线段，所需轮次为 $O(\sqrt{n}\log n)$。由于任意两个变形虫机器人数量相同的连通结构都可以分别重构为一条线段，且每次重构都可逆，因此我们可以立即得出，任意结构都能在相同的渐近时间界内重构为另一个任意结构。因此，我们可以在 $O(\sqrt{n}\log n)$ 轮内实现通用重构。这回答了一个开放问题：在不作额外假设的情况下，能否在次线性时间内实现大规模重构 [14]。我们还提出了一种算法，可在常数时间内将螺旋结构重构为一条线段。这一结果表明，路径的非平凡子类可以在模型内极快地完成重构。我们希望这能成为迈向多对数时间甚至常数时间通用重构的基础性一步；后者仍是一个核心开放问题。

总的来说，我们的结果表明，联合移动能够大幅提升可编程物质结构的重配置效率，也就是将一种结构重配置为另一种结构所需的速度。尽管原始 amoebot 模型受到基于直径的下界限制，但正如我们所展示的，协调并行运动能够实现显著更快的全局重配置。

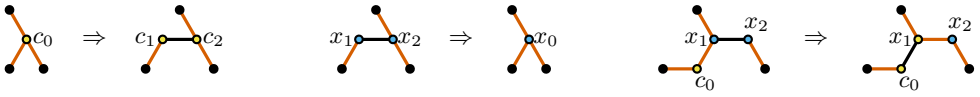
论文结构。 第 2 节介绍带有联合移动的几何 Amoebot 模型，并正式定义重配置问题以及本文所考虑的结构类别。第 3 节介绍一组基本运动原语，作为我们算法的构建模块。第 4 节给出将单调结构、直方图结构和星形凸结构等特殊类别的结构重配置为更简单的标准形式（例如线段）的算法。第 5 节介绍我们的主要结果：一种通用重配置算法，能够在次线性时间内将任意连通结构重配置为线段。第 6 节介绍将螺旋结构重配置为线段的算法，其中包括一种常数时间解法。最后，第 7 节讨论开放问题。

2 模型与问题陈述

本节将正式介绍模型和问题陈述。第 2.1 节介绍带有联合移动的几何 Amoebot 模型。第 2.2 节定义重配置问题，以及本文所考虑的变形虫机器人结构类别。

2.1 带有联合移动的几何 Amoebot 模型

本节介绍带有联合移动扩展 [14] 的几何 Amoebot 模型 [8, 9] 。变形虫机器人结构由放置在无限正则三角网格图 $G_\Delta = (V_\Delta, E_\Delta)$ 上的 n 个变形虫机器人组成（见图 1 左侧）。每个变形虫机器人占据一个节点，或两个相邻节点以及它们之间的边。若变形虫机器人占据一个节点，我们称其为收缩态；否则称其为扩展态。G_Δ 中的每个节点至多由一个变形虫机器人占据。相邻的变形虫机器人通过键相连。



（**a**）扩展。 （**b**）收缩。 （**c**）交接。

图 2 移动。（a）和（c）中的黄色变形虫机器人表示变形虫机器人 c_0。（b）和（c）中的蓝色变形虫机器人表示变形虫机器人 x_0。

令 C 表示收缩状态变形虫机器人所占据的节点集合， X 表示扩展状态变形虫机器人所占据的节点集合， A 表示扩展状态变形虫机器人所占据的所有边的集合， B 表示所有键的集合。我们将变形虫机器人结构的连通图定义为图 $S = (C \cup X,\ A \cup B)$。 A 中的边两两不相交。我们假设初始时 S 是连通的。我们为 $A \cup B$ 中的每条边指定一个几何方向。我们只允许与三角网格的轴平行的方向 G_Δ （见图 1 右侧）。变形虫机器人结构由其连通图及方向的指定所定义。为简便起见，我们将每个变形虫机器人结构都用其连通图表示。

我们假设采用完全同步激活模型，即时间划分为同步轮次，且每个变形虫机器人在每一轮中都处于激活状态。在每一轮中，每个变形虫机器人都可以执行一次移动，该移动分为两个步骤完成。设 $S_i = (C_i \cup X_i, A_i \cup B_i)$ 为该轮开始时的变形虫机器人结构。

在**第一步**中，变形虫机器人按如下方式从 S_i 移除键。每个变形虫机器人都可以决定在 S_i中释放其当前关联键的任意子集。当且仅当某个关联的变形虫机器人释放了某条键时，该键才会被移除。请注意，我们只移除键（即 B_i中的边），而不移除占用边（即 A_i中的边）。令 $R_i \subseteq B_i$ 为剩余键的集合， $S'_i = (C_i \cup X_i, A_i \cup R_i)$为由此得到的变形虫机器人结构。我们要求 S'_i 是连通的，否则不连通的部分可能会彼此分离。我们称当且仅当 S'_i 不连通时发生**连通性冲突**。每当发生连通性冲突时，变形虫机器人结构就会转变为未定义状态，因此我们无法对该结构作出任何判断。

在**第二步**中，每个变形虫机器人都可以执行以下移动之一。占据 $c_0 \in C_i$ 的收缩状态变形虫机器人 c 可以沿网格的一条轴**扩展** （见图 2a）。为此，我们将 c_0 替换为两个节点 c_1, c_2 和一条边 $a = \{c_1, c_2\}$，并为 a指定一个方向。对于每条与 c 关联的键 $b \in R_i$，我们将 c_0 替换为其中一个新节点。请注意，关联键的方向不会改变。因此，所有相连的变形虫机器人都会随着扩展的变形虫机器人一起移动。扩展状态变形虫机器人可以通过反向执行扩展操作来进行收缩（见图 2b）。

此外，变形虫机器人对还可以执行独立的**交接**，如下所示（见图2c）。考虑一个占据 $c_0 \in C_i$ 的收缩状态变形虫机器人 c ，以及一个占据$x_1, x_2 \in X_i$和 $a = \{x_1, x_2\} \in A_i$的扩展状态变形虫机器人 x ，二者由键 $b = \{c_0, x_1\} \in R_i$连接。直观地说，我们希望交换扩展状态变形虫机器人与该键的关联。更准确地说，(i) 变形虫机器人 c 变为扩展态，并占据节点 c_0、x_1和边 b；(ii) 变形虫机器人 x 变为收缩态，并占据节点 x_2；(iii) 边 a 变为键。

移动结束时，令 C_{i+1} 表示收缩状态变形虫机器人占据的节点集合， X_{i+1} 表示扩展状态变形虫机器人占据的节点集合， A_{i+1} 表示扩展状态变形虫机器人占据的所有边的集合，R'_i 表示所有键的集合。令 $S''_i = (C_{i+1} \cup X_{i+1},\ A_{i+1} \cup R'_i)$ 为所得的连通图。我们要求 S''_i 是 G_Δ 的子图，并符合所有边的方向（即 $A_{i+1} \cup R'_i$）。当且仅当 S''_i 不是 G_Δ 的子图时，我们称发生了**碰撞**，也就是说，变形虫机器人无法映射到三角

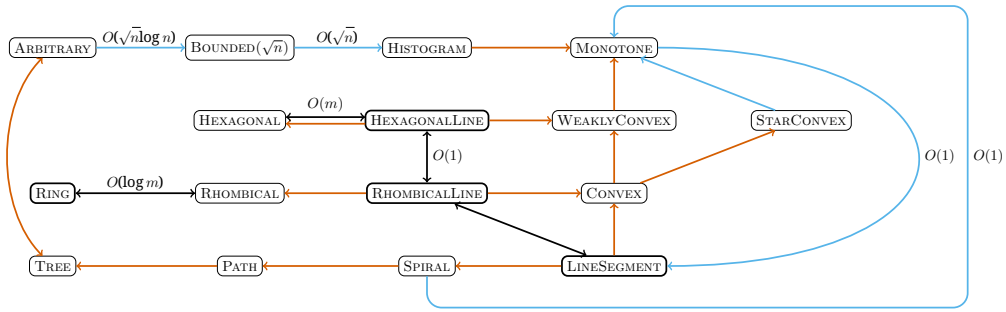


图3 结果。边框较粗的类表示规范类。红色箭头指向超类，因此表示平凡解。请注意，我们没有展示所有超类。黑色箭头表示 Padalkin 等人 [14]的结果。请注意，他们的超常数算法模拟了来自 [5, 12]的算法。他们定义了菱形和六边形元模块。Rhombical（Hexagonal）包含所有由菱形（六边形）元模块构成的变形虫机器人结构。RhombicalLine（HexagonalLine）包含所有由一列菱形（六边形）元模块构成的变形虫机器人结构。Ring 包含构成环的规范变形虫机器人结构（详情见 [5] ）。m 表示元模块的数量。蓝色边表示我们的结果。请注意，我们将有界结构重配置为直方图的算法模拟了来自 [4]的算法。

网格上，或者多个变形虫机器人映射到同一个节点上。每当发生碰撞时，变形虫机器人结构就会转变为未定义状态，以致我们无法对该结构作出任何陈述。作为算法设计者，我们有责任确保不会发生碰撞。在下一轮中，我们继续采用 S_i'' 的邻接闭包$S_{i+1} = (C_{i+1} \cup X_{i+1},\ A_{i+1} \cup B_{i+1})$，即在变形虫机器人之间添加键，直到所有相邻的变形虫机器人都相连。

本文假设存在一个集中式调度器。调度器始终知晓变形虫机器人结构的当前状态。在每个同步轮次开始时，调度器都会为每个变形虫机器人决定要释放哪些键，以及执行哪种移动。由于本文着重探讨该模型中次线性时间动力学的可行性（即不作额外假设，以区别于先前采用元模块的工作），我们将分布式解决方案的设计留待未来研究。

2.2 问题陈述与变形虫机器人结构的类别

本文研究同规模的两类变形虫机器人结构之间的重配置问题 A 和 B 。该问题的目标是设计一种算法，使其能够针对每个变形虫机器人结构 $S \in A$ ，计算出一个调度，将 S 重构为任意一个变形虫机器人结构 $S' \in B$。请注意，算法可能只会将 A 中的变形虫机器人结构映射到 B 的一个子集，而非整个类别。因此，一般而言，无法通过使用同一算法并反转调度来解决反向问题。下文将讨论以下类别（见图3，其中也展示了我们的结果）。我们假设这些类别中的所有结构都只由收缩态变形虫机器人组成。请注意，每个类别在平移、旋转和反射下均保持封闭。

任意：所有结构。

树：构成树的结构。请注意，由于每个结构都有一棵生成树，因此此类别等同于任意。

路径：构成一条路径的结构。

Spiral：由 $120°$ 个拐角组成的简单螺旋结构。请注意，单条线段就是只有一段的螺旋。

LineSegment：形成线段的结构。
Monotone：我们将变形虫机器人结构与平行于x轴的直线的交称为 x-截面。注意， x-截面不一定连通。我们类似地定义 y-和z-截面 。特别地，我们也将 x-截面称为行，并将 y-截面称为列。当且仅当每个 x-截面都连通时，我们称变形虫机器人结构为 x-*Monotone* 。我们类似地定义y-和 z-Monotone。当且仅当变形虫机器人结构是x-Monotone、 y-Monotone或 z-Monotone时，我们称其为*Monotone* 。

直方图：我们将平行于 x轴的变形虫机器人线段称为 x-线段。我们类似地定义 y-和 z-线段 。当且仅当变形虫机器人结构由一个 x-线段以及附着在该 x-线段一侧的 y-线段组成，或可由这样的结构通过反射和旋转得到时，我们称该变形虫机器人结构构成直方图 。注意，每个直方图都是Monotone。

有界结构(k)：最多包含 k 个非空的 x、 y 或 z 线段的结构。
凸：当且仅当对于每一对变形虫机器人 u， v，它们之间的所有最短路径都属于该变形虫机器人结构时，我们称该变形虫机器人结构为 凸。
弱凸：当且仅当对于每一对变形虫机器人 u， v，它们之间至少有一条最短路径属于该变形虫机器人结构时，我们称该变形虫机器人结构为 弱凸。
星凸：当且仅当存在一个变形虫机器人u ，使得对于每个变形虫机器人 v， u 与 v 之间的所有最短路径都属于该变形虫机器人结构 [6]时，我们称该变形虫机器人结构为 星凸。

重配置问题的另一个有趣变体是在某个类 A内进行重配置。该变体的目标是设计一种算法，对于任意两个变形虫机器人结构S， $S' \in A$ （其中变形虫机器人的数量相同），计算出一个将 S 重配置为S'的调度。当我们针对 Arbitrary 类解决该问题时，就得到了通用重配置算法。设计这类算法的一种常见方法是，在重配置过程中使用一个规范结构 C 作为中间结构（例如， [4, 5, 12]）。给定从S 到 C以及从 S' 到 C的调度，我们可以先执行从 S到 C 的调度，然后执行从 S' 到 C的反向调度，从而得到从 S 到 S' 的调度。在本文中，我们将线段用作规范结构。因此，我们的重点是解决各个类与 LineSegment 类之间的重配置问题。我们的主要成果是一个需要 $O(\sqrt{n}\log\ n)$ 轮的通用重配置算法，以及一个能在 $O(1)$ 轮内将 Spiral 重配置为直线的算法。更多结果见图3。

3 预备知识

本节将介绍基本的运动原语：隧道原语、剪切原语、平行四边形原语、三角形原语和梯形原语。它们是我们重构算法的核心组成部分。

在隧穿原语中（最初定义于 [10], ），我们通过不断进行交接，使一条由变形虫机器人组成的链沿路径移动。[1] 算法 1 展示了每个变形虫机器人执行的算法。我们将使用这一原语，将一个变形虫机器人从一端“隧穿”到

[1] 在以往的工作中，该原语是针对生成森林定义的，因此称为生成森林原语。由于我们只需要路径这一较简单的情况，因此一般情况请参见 [7, 10] 。

算法 1 隧穿原语 [7]

```
如果 处于收缩态则
    如果 链中的最后一个变形虫机器人则
        收缩
    end
    else
        与前驱执行交接
    end
end
else
    如果 是链的第一个变形虫机器人那么
        沿路径扩展
    end
    else
        与后继者进行交接
    end
end
```



图4 隧穿原语。在每一轮中，颜色相同且相邻的变形虫机器人执行交接。灰色节点未被占据。在所有图中，两个子图之间的简单箭头表示在建立邻接闭包后执行一轮的第一步（即移除键），双向箭头表示执行一轮的第二步（即移动），除非另有说明。请注意，只有完整执行的轮次（即两个步骤均已执行）才计入轮次总数。

另一端，同时不改变其余链的结构。一般而言，将该变形虫机器人隧穿通过整条链最多可能需要 $O(n)$ 轮次。我们将由收缩态和扩展态变形虫机器人交替组成的链称为交替链。对于交替链，该原语只需 3 轮（见图4）。注意，该原语实际上会使所有变形虫机器人沿路径移动，但不会改变它们的顺序。然而，由于变形虫机器人彼此相同，其结果等同于一次真正的隧穿。根据构造，我们得到以下引理。

▶**引理 1**。 在交替链上，隧穿原语需要3 轮才能使一个变形虫机器人穿过该链。

我们在变形虫机器人结构的子结构上执行所有其余原语。变形虫机器人结构的其他部分可能在特定的变形虫机器人处与该子结构相连，我们称这些变形虫机器人为**连接点**。每个原语都有两个连接点。由于这些部分在没有该子结构的情况下不一定相连，因此我们必须始终保持该子结构处于连通状态。因此，我们必须始终保持该子结构处于连通状态。除了剪切原语外，我们保持这两个连接点之间的相对位置不变。

在剪切原语中，我们将一条由变形虫机器人组成的线段剪切到另一条轴上。[2] 我们强调，这并非旋转线段，因为变形虫机器人结构中与线段相连的部分不会随线段旋转。

[2] 该原语类似于 [14] 的重新对齐原语。事实上，这两个原语使用相同的中间结构。因此，我们可以在这两个原语的所有结构之间进行切换。

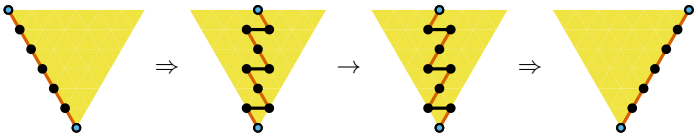


图5 剪切原语。蓝色变形虫机器人表示连接点。黄色区域表示从底部连接点的视角来看，该原语在其中执行的三角形。

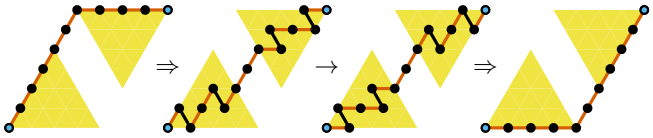


图6 平行四边形原语。蓝色变形虫机器人表示连接点。

直线的端点用作连接点。

我们按如下方式实现该原语（见图5）。该原语由两轮组成。在第一轮中，我们通过扩展每隔一个变形虫机器人，将直线剪切到一半。在第二轮中，我们镜像第一轮，将线段剪切到新轴上。请注意，从连接点的视角来看，剪切原语是在由线段的初始位置和最终位置所定义的三角形内执行的（见图5中的黄色区域）。根据构造，我们得到如下引理。

▶**引理 2.** 我们对剪切原语的实现需要2 轮次。

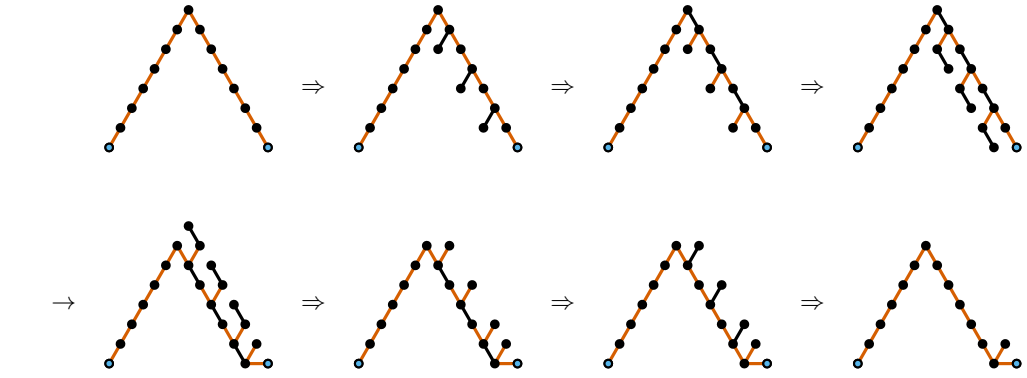
▶ 注记3. 剪切原语可以推广到由收缩态amoebot构成的平行四边形。

考虑一个平行四边形，其中只有与一个钝角相邻的两条边上有变形虫机器人。令 ℓ_1 和 ℓ_2 表示平行四边形的边长，不失一般性， $\ell_1 \leq \ell_2$。位于锐角处的变形虫机器人充当连接点。平行四边形原语的目标是在不离开平行四边形且始终不改变连接点相对位置的情况下，将变形虫机器人移动到另外两条边上。注意，这两条边所需的变形虫机器人数量与最初被占据的两条边相同。

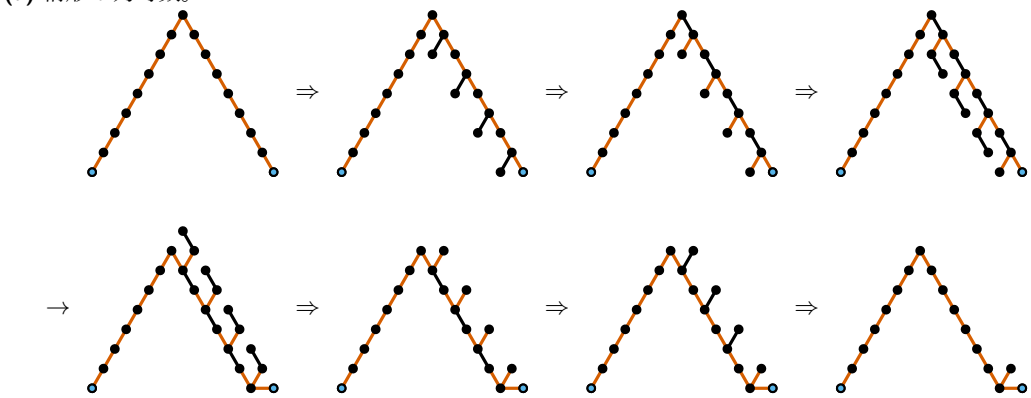
我们按如下方式实现该原语（见图 6）。在平行四边形的两条边上，分别从连接点开始，对前 $\ell_1 + 1$ 个变形虫机器人应用剪切原语。注意，如果平行四边形是菱形，那么两条边上第 $(\ell + 1)$ 个变形虫机器人是同一个，并且会参与两个剪切原语。这是可行的，因为两个剪切原语所需的移动相匹配。根据构造，我们得到以下引理。

▶**引理4.** 我们对平行四边形原语的实现需要2 轮次。

考虑一个边长为 ℓ 的等边三角形，其中只有两条边（不失一般性地，腰）被变形虫机器人占据。与未占据边（即底边）相邻的角上的变形虫机器人充当连接点。三角形原语的目标是在不离开三角形的前提下占据底边和其中一条腰，并且始终不改变连接点的相对位置。注意，由于该三角形是等边三角形，任意两条边所需的变形虫机器人数量相同。



(a) 情形 ℓ 为奇数。



(b) 情形 ℓ 为偶数。

图 7 三角形原语的第一阶段。蓝色的变形虫机器人表示连接点。

我们无法简单地将一条腰剪切到底边，因为这会使三角形内的变形虫机器人断开连接（例如，直接将一条腰剪切到底边），或者改变连接点的相对位置（例如，先将一条腰剪切到另一条腰，使两条腰相邻，然后再将其剪切到底边）。相反，我们按如下方式实现该原语。不失一般性地，假设腰与 y 轴和 z 轴对齐。我们的实现包含 4 个阶段。我们先概述这些阶段。第一阶段，我们迭代地将三角形的大小（即其边长）缩小 1，直到 $\ell = 1$ 或者存在一个偶数的 k，使得 $\ell = 4k + 1$。第二阶段，我们占据底边。不过，为此我们要从两条腰上移出。第三阶段，我们占据三角形内所有与底边相邻的节点。第四阶段，我们将这些变形虫机器人移到其中一条腰上。

考虑**第一阶段**，即对三角形进行迭代缩减（见图 7）。我们从顶部开始，将一条边上的变形虫机器人两两配对；若 ℓ 为奇数，则最后 2 个变形虫机器人除外；若 ℓ 为偶数，则最后 1 个变形虫机器人除外。然后，我们对每一对使用隧穿原语，以填充这条边相邻的节点。我们将多出的两个变形虫机器人放在原始边的底部。这一点在第四阶段会变得重要。现在暂且忽略它们。

当 $\ell = 1$ 或存在一个偶数 k 使得 $\ell = 4k + 1$时，我们停止。如果满足前一种情况，就直接进入第四阶段。否则，我们继续进行第二阶段。

考虑**第二阶段**，即底边的占据情况（见图 8）。在每条腿中，我们对两个子段应用剪切原语。令 $(v_1, \ldots, v_\ell = v_{4k+2})$ 表示一条腿上从底部到顶部的变形虫机器人。在左边腿中，我们对 $(v_1, \ldots, v_{2k+1})$ 进行剪切，使其与 x-轴对齐；并对 $(v_{2k+1}, \ldots, v_{3k+1})$ 进行剪切，使其与 z-轴对齐。在右边腿中，我们剪切

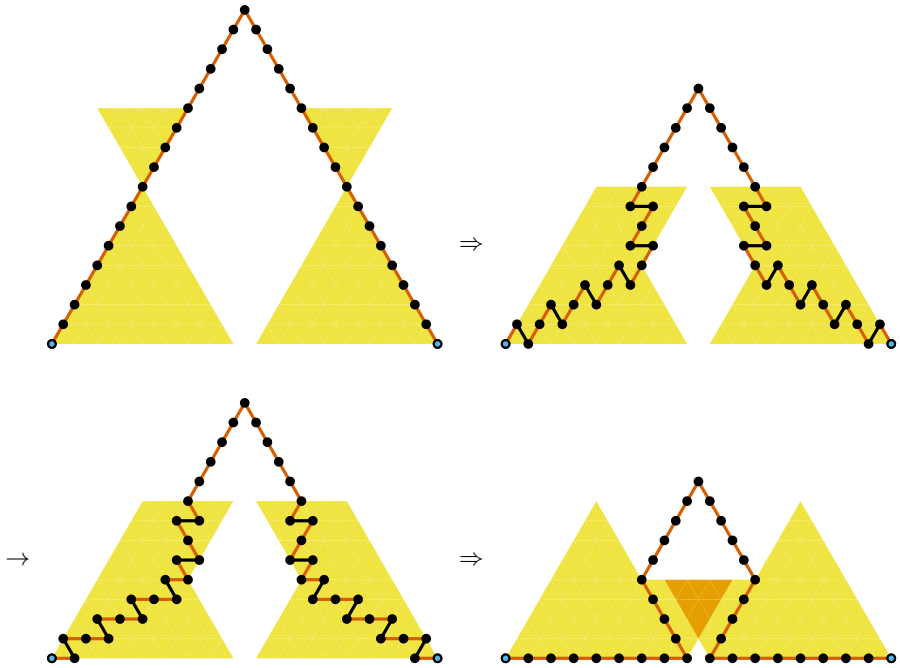


图 8 边长为 $\ell = 17$ 的三角形上三角形原语的第二阶段。注意，$k = 4$ 成立。蓝色的变形虫机器人表示连接点。

$(v_1, \ldots, v_{2k+1})$，使其与 x-轴对齐；并对 $(v_{2k+1}, \ldots, v_{3k+1})$ 进行剪切，使其与 y-轴对齐。两条腿中子段的长度经过选择，使它们的水平移动相互抵消。这意味着其余变形虫机器人会笔直向下移动，并且连接点之间的相对位置不会改变。由于 $\ell = 4k + 1$，阶段结束后底边上的所有节点都被占据。此外，剩余的变形虫机器人在底边上方形成一个缺少顶角的菱形。我们继续进行下一阶段。

考虑**第三阶段**，即占据与底边相邻的节点（见图 9）。由于底边已被占据，我们可以断开顶部的菱形，从而得到两条“臂”。我们对两条臂都使用剪切原语，将它们移到底边。我们先移动较短的臂。然后，通过一次扩张与收缩，将较长的臂移至中心。这样便可对其进行剪切，而不会离开三角形。现在，底边及其所有相邻节点都已被占据。注意，第一阶段中多出的变形虫机器人也满足这一点。接下来进入下一阶段。

考虑**第四阶段**，即占据其中一条腰（见图 10）。在此，我们只需对与底边相邻的变形虫机器人再次应用剪切原语，将它们移至其中一条腰。根据构造，我们得到以下引理。

▶**引理 5。** 三角形原语的实现需要 $O(1)$ 轮次。

最后，考虑一个较短底边和腰均由变形虫机器人占据的梯形。位于锐角处的变形虫机器人充当连接点。给定较长底边上的一个节点。我们将其称为起点。梯形原语的目标是在不离开梯形且始终不改变连接点相对位置的情况下，占据较长底边以及一条从起点开始、位于两条底边之间的路径，该路径最多可由 2 条线段组成。请注意，较短底边和两条腰所需的变形虫机器人数量，与较长底边和一条腰所需的数量相同。此外，

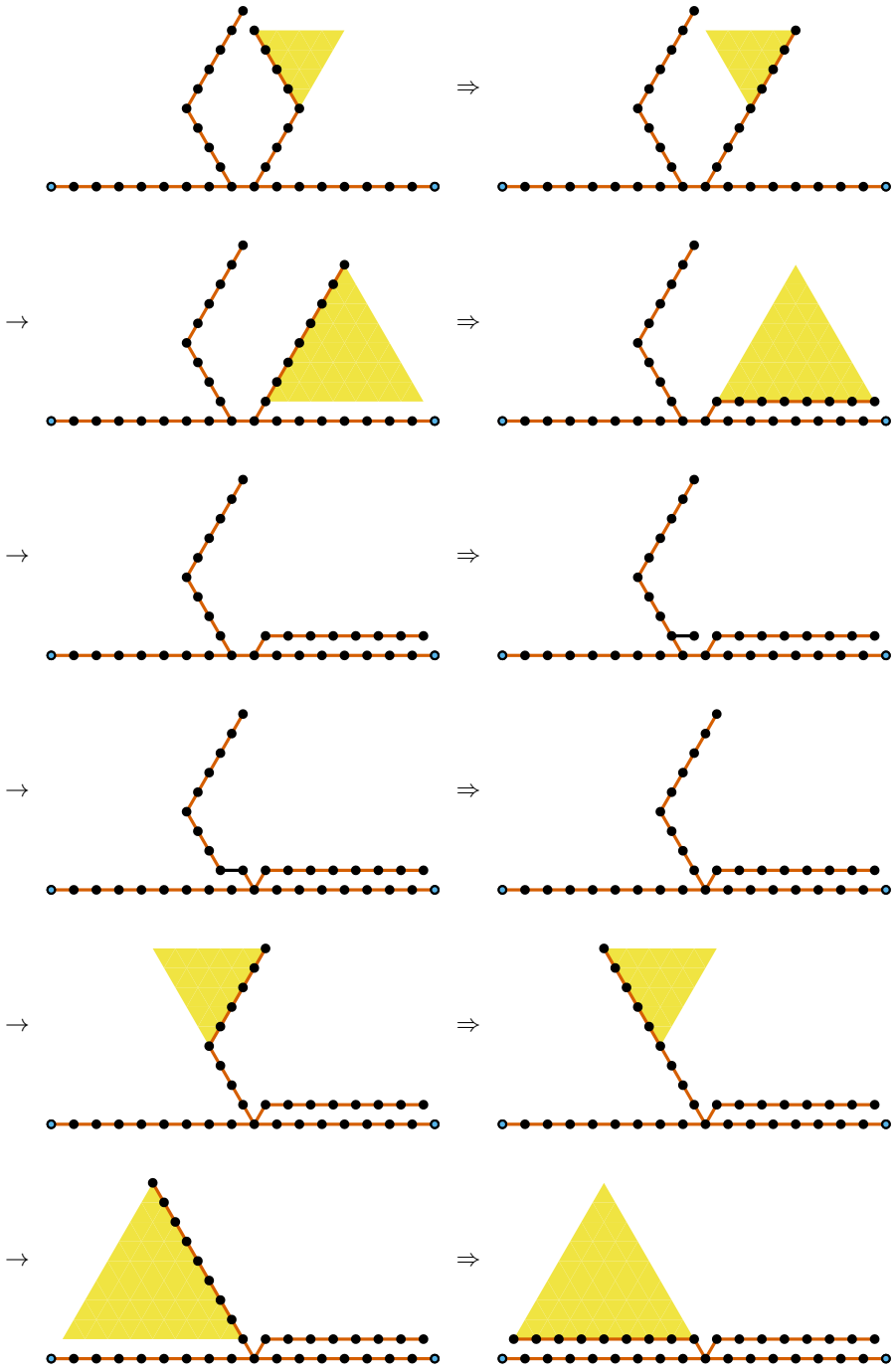


图 9 三角形原语的第三阶段。蓝色的变形虫机器人表示连接点。前两个和最后两个双向箭头表示 2 轮。

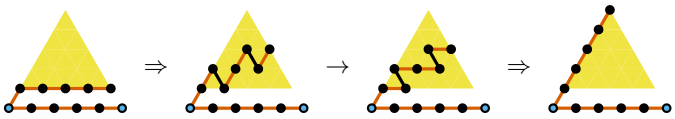


图 10 三角形原语的第四阶段。蓝色变形虫机器人表示连接点。

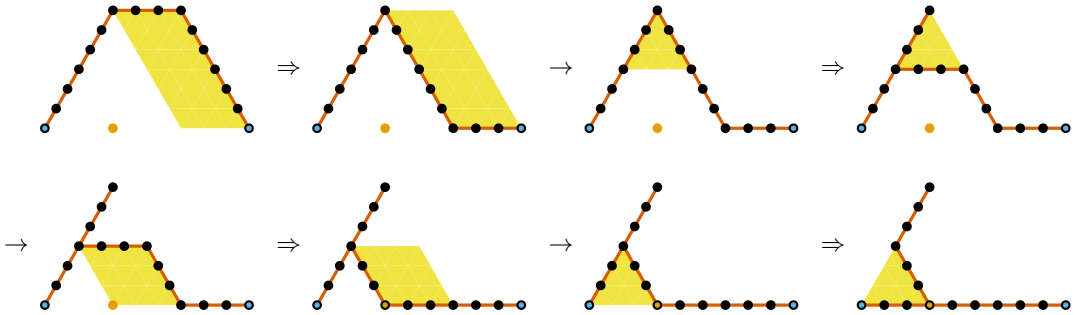


图11 梯形原语。橙色节点/变形虫机器人表示起点。蓝色变形虫机器人表示连接点。每个双向箭头表示一个由 2 轮组成的阶段。

两腰之间的路径所需的变形虫机器人数量与一条腰上的数量相同。因此，此次重构所需的变形虫机器人数量恰好满足要求。

我们按如下方式实现该原语（见图11）。我们的实现包含 4 个阶段。在**第一阶段**，我们将梯形分成一个三角形和一个平行四边形，使起点位于三角形内。然后，我们对平行四边形应用平行四边形原语。这会部分占据较长底边。接下来需要处理由此形成的三角形。

在**第二阶段**，我们区分两种情况。如果起点是三角形的一个角，我们就在该三角形上应用三角形原语，使得完成时，被占据的边从起点开始。这样便占据了较长底边剩余的部分。此外，被占据的边形成一条从起点出发、连接两条底边的路径（即，该路径由一条线段组成）。此时，我们终止该原语。

否则，也就是说，如果起点不是三角形的角，我们就将三角形分成两个三角形和一个平行四边形，使起点成为下方三角形和平行四边形的一个角。然后，我们在上方三角形上应用三角形原语。我们得到一个梯形（由下方三角形和平行四边形组成），其一个角上带有一条附加边。

在**第三和第四阶段**，我们在保持附加边相连的同时重复第一和第二阶段。不过，这次我们可以终止，因为起点是三角形的一个角。注意，第二和第四阶段得到的腰构成一条连接两条底边的路径，该路径从起点开始（即，该路径由2条线段组成）。根据构造，我们得到以下引理。

▶**引理6**。 我们的梯形原语实现需要 $O(1)$ 轮次。

▶注7. 利用剪切原语，我们可以反射从起点开始、位于两条底边之间的路径。

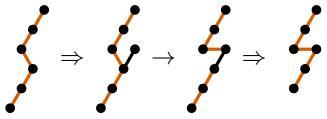


图 12 Monotone2LineSegment 算法的预处理阶段。

4 单调结构

我们的通用重配置算法和常数时间的 Spiral2Line 算法都使用单调结构作为中间结构。在第 4.1 节中，我们展示如何将单调结构重构为一条线段。在第 4.2 节中，我们调整该算法，使其能在特定区域内将直方图重构为一条线段。这样便可以将星形凸结构重构为单调结构。

4.1 Monotone2LineSegment 算法

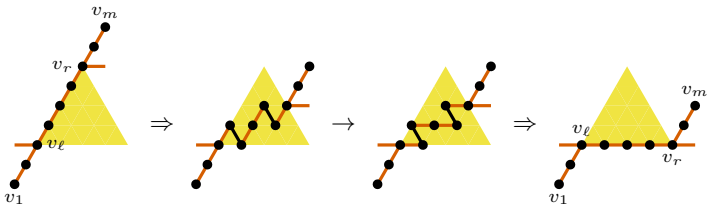
本节介绍我们的 Monotone2LineSegment 算法。不失一般性，我们假设变形虫机器人结构是 y-单调的。根据定义，每一列都是一个 y-线段。我们首先移除相邻列之间的所有键，只保留一条任意的水平键。如果不存在这样的键，我们就在预处理阶段将右列向上移动一个位置（见图 12）。

令 (v_1、.....、 v_m) 表示该列中从底部到顶部的变形虫机器人。令 v_ℓ (v_r) 表示与左（右）列相连的变形虫机器人（如果存在），否则为 $v_\ell = v_1$ ($v_r = v_m$)。每一列都应用以下规则，直到 $m = 1$ （见图 13）。

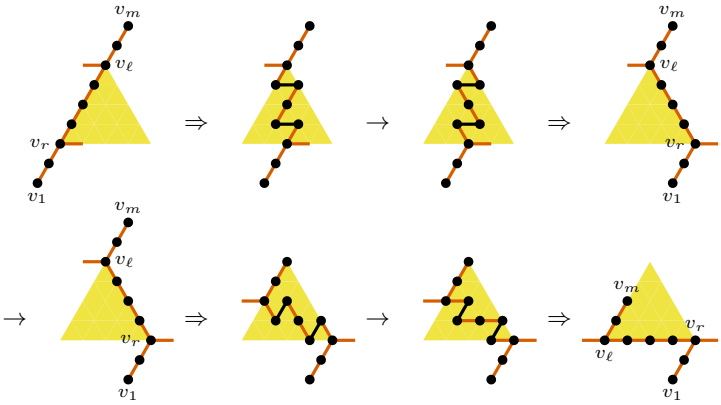
1. 如果 $\ell < r$，我们将子段（$v_\ell, \ldots, v_r$）剪切，使其与 x 轴对齐（见图 13a）。我们得到 $r - \ell + 1$ 个新列。第一列由（$v_1, \ldots, v_\ell$）组成。对于所有 $i \in \{2, \ldots, r - \ell\}$，第 i 列由 $v_{\ell-1+i}$ 组成。最后一列由（$v_r, \ldots, v_m$）组成。对于这些列中的每一列，$\ell = r$ 成立。请注意，如果 $\ell = 1 \wedge r = m$，则所有生成的列各自都只包含一个变形虫机器人。

2. 如果 $\ell > r$，我们将子段（$v_r, \ldots, v_\ell$）剪切两次，先使其与 y 轴对齐，再使其与 x 轴对齐（见图 13b）。我们得到 $\ell - r + 1$ 个新列。第一列由（$v_\ell, \ldots, v_m$）组成。对于所有 $i \in \{2, \ldots, \ell - r\}$，第 i 列由 $v_{\ell+1-i}$ 组成。最后一列由（$v_1, \ldots, v_r$）组成。对于这些列中的每一列，$\ell = r$ 成立。请注意，如果$\ell = m \wedge r = 1$，则所有生成的列各自都只包含一个变形虫机器人。

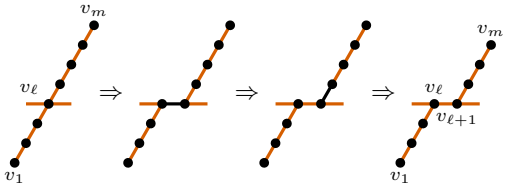
3. 若 $\ell = r$，我们应用以下子规则。
   a. 如果 $1 < \ell < m$，我们按如下方式将该列拆分为两列（见图 13c）。首先，v_ℓ 水平扩展。其次，v_ℓ 和 $v_{\ell+1}$ 进行交接。最后，$v_{\ell+1}$再次收缩。左列由（$v_1, \ldots, v_\ell$）组成。右列由（$v_{\ell+1}, \ldots, v_m$）组成。b. 如果 $\ell = 1$ 和 m 为奇数，我们按照规则 3a 类似的方式将该列拆分为两列（见图 13d）。左列由 v_1组成。右列由（$v_2, \ldots, v_m$）组成。c. 如果 $\ell = m$ 和 m 为奇数，我们按照规则 3a 和 3b 类似的方式将该列拆分为两列（见图 13e）。左列由 v_m组成。右列由（$v_2, \ldots, v_{m-1}$）组成。d. 如果 $\ell = 1$ 和 m 为偶数，我们按如下方式将子段（$v_\ell, \ldots, v_m$）拆分为两列，每列分别包含$m/2$ 个变形虫机器人（见图 13f）。对于每个 $i \in \{1, 3, \ldots, m - 1\}$,$v_i$，我们并行执行以下过程。在第一轮中，每个 v_i 扩展



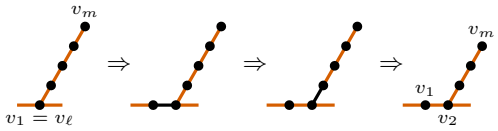
(a) 规则 1（$\ell < r$）。



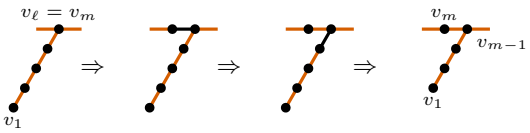
(b) 规则 2（$\ell > r$）。



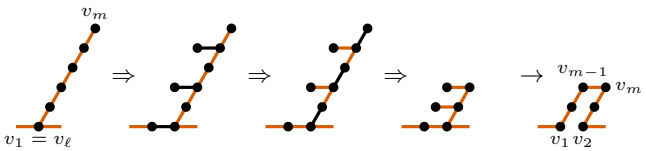
(c) 规则 3a（$\ell = r$ 和 $1 < \ell < m$）。



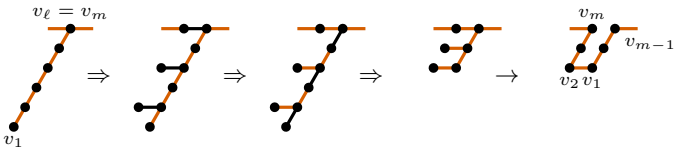
(d) 规则 3b（$\ell = r$、 $\ell = 1$ 和 m 为奇数）。



(e) 规则 3c（$\ell = r$、 $\ell = m$ 和 m 为奇数）。



(f) 规则 3d（$\ell = r$、 $\ell = 1$ 和 m 为偶数）。



(g) 规则 3e（$\ell = r$、 $\ell = m$ 与 m 均为偶数）。

图 13 Monotone2LineSegment 算法中使用的规则。

水平地。在第二轮中，每个 v_i 与 v_{i+1}执行一次交接。在第三轮中，每个 v_{i+1} 收缩。我们得到含有 $m/2$ 个变形虫机器人的两列。左列由（v_1、v_3、……、v_{m-1}）组成。右列由（v_2、v_4、……、v_m）组成。对于后续规则的应用，我们将同一列内的所有变形虫机器人用键连接起来，并移除这两列之间除最顶部水平键以外的所有键。请注意，这不需要额外的一轮。e. 若 $\ell = m$ 与 m 为偶数，我们类似于规则 3d（见图 13g），将子段（v_ℓ、……、v_m）分别拆分为各含$m/2$ 个变形虫机器人的两列。左列由（v_2、v_4、……、v_m）组成。右列由（v_1、v_3、……、v_{m-1}）组成。对于后续规则的应用，我们将同一列内的所有变形虫机器人用键连接起来，并移除这两列之间除最底部水平键以外的所有键。请注意，这不需要额外的一轮。

我们得到如下定理。

▶**定理 8**。 *Monotone2LineSegment* 算法在 MONOTONE轮内将一个变形虫机器人结构重构为 LINESEGMENT 结构，共需 16轮。

证明。 首先，分别考虑预处理阶段和每条规则。我们证明，这些步骤的执行不会造成任何碰撞。在预处理阶段，没有变形虫机器人离开其所在的列。因此，不会与其他列中的变形虫机器人发生碰撞。在每条规则中，该列将其相邻的列向左和向右推移，同时创建新的列。这可以避免与其他列发生碰撞。最后，注意列内不可能发生碰撞，因为每一列始终保持连通。最初，根据定义这一点成立，因为变形虫机器人结构是 Monotone 的。此后，根据预处理阶段和所有规则的构造方式，这一点仍然成立。

为便于分析，我们将规则 1 和规则 2 拆分：规则 1a 包含所有 $\ell < r \wedge (\ell \neq 1 \vee r \neq m)$ 情形，规则 1b 包含所有 $\ell < r \wedge (\ell = 1 \wedge r = m)$ 情形，规则 2a 包含所有 $\ell > r \wedge (\ell \neq m \vee r \neq 1)$ 情形，规则 2b 包含所有 $\ell > r \wedge (\ell = m \wedge r = 1)$ 情形。请注意，我们并未改变规则的执行方式。

现在，考虑任意一个变形虫机器人。我们按如下方式为该变形虫机器人构造状态图（见图 14）。变形虫机器人的状态由需要在其所在列上应用的规则定义。这包括不存在可应用规则的状态。显然，该状态也是终止状态。当且仅当变形虫机器人在其当前列上应用规则 x 后，最终处于某一列，而我们需要在该列上应用规则 y 时，我们才从规则 x 向规则 y 添加一条边。图 14 展示了所得的图。初始时，变形虫机器人可以处于包括终止状态在内的任意状态。然而，请注意，该图是无环的。因此，变形虫机器人至多经过 4 条规则后就会到达终止状态。

由于预处理步骤只需要 2 轮，每个变形虫机器人至多参与 4 条规则，每条规则只需要 $O(1)$ 轮，因此算法将在 $O(1)$ 轮后终止。更准确地说，状态图中的最长路径加上预处理阶段需要 16 轮。由于处于终止状态的任意变形虫机器人都是其所在列中唯一的变形虫机器人，并且只有水平键，因此当所有变形虫机器人都终止后，我们将得到一条线段，即算法是正确的。◀

4.2 StarConvex2Monotone 算法

在本节中，我们将星形凸结构重构为单调结构。为此，我们首先需要考虑在下文将定义的特定区域内将直方图重构为线段。请注意，若无此限制，我们可以应用

16 通过联合移动对可编程物质进行次线性时间重构

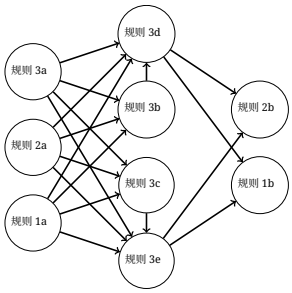


图 14 状态图。未显示终止状态。其他任何状态都有一条边通向终止状态。

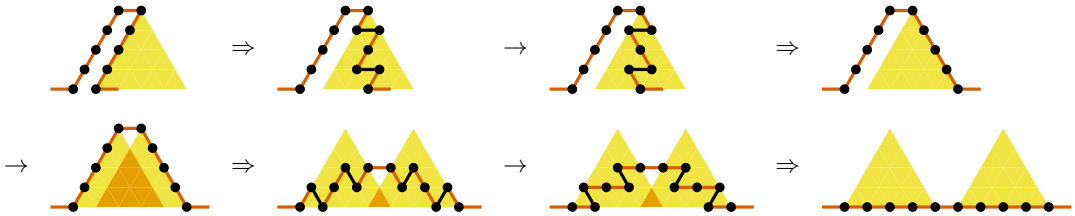


图 15 规则 4。我们省略第一阶段，即应用规则 3d（见图 13f），仅展示第二和第三阶段。

Monotone2LineSegment 算法将直方图重构为线段，因为直方图是单调的。

不失一般性，考虑一个由 x-线段以及连接在顶部的 y-线段组成的直方图。我们的目标是将其重构为一条线段，同时确保没有变形虫机器人移动到经过 x-线段上最左侧变形虫机器人的 x-轴和 y-轴之外。将星形凸结构重构为单调结构时，我们将使用最左侧的变形虫机器人作为连接点。

我们对 Monotone2LineSegment 算法作如下调整。首先，选择沿 x-线段的键作为列之间的水平键。其次，将规则 1、规则 2 和规则 3d 合并为一条新规则：

4. 如果 $\ell = 1$ 且 m 为偶数，我们分别将该列拆分为 m 列，每列包含一个变形虫机器人，具体如下（见图 15）。该规则包含 3 个阶段。第一阶段，应用规则 3d。第二阶段，在右列应用规则 2 的第一个剪切原语。第三阶段，在左列应用规则 1，并在右列同时应用规则 2 的第二个剪切原语。

我们得到如下引理。

▶**引理 9.** *Histogram2LineSegment* 算法，即改编后的 *Monotone2Line* 算法，在 $O(1)$轮次内将直方图 结构重配置为LineSegment 结构，且不使任何变形虫机器人移动到经过最左侧变形虫机器人的 x 轴和 y 轴之外。

证明。 正确性和运行时间主要由定理 8 得出。注意，规则 4 不会导致任何碰撞，因为它是规则 1、2 和 3d 的组合。剩下只需证明，我们不会将任何变形虫机器人移动到穿过最左侧变形虫机器人的 x-轴和 y-轴之外。为此，我们论证每条规则都只会将变形虫机器人向右推动（从而使其远离穿过最左侧变形虫机器人的 y-轴），且不会将任何变形虫机器人移动到穿过最左侧变形虫机器人的x-轴下方。注意，我们只需考虑规则 3b 和 4，因为最初每一列中的所有水平键都是该列最底部的键，而这两条规则

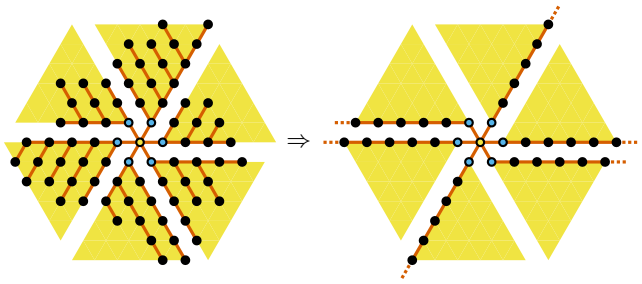


图 16 星形凸结构中的直方图。黄色变形虫机器人表示中心变形虫机器人。蓝色变形虫机器人表示 Histogram2LineSegment 算法所用的连接点。黄色三角形分别表示每个实例中不会移出任何变形虫机器人的区域。双向箭头表示多个轮次。

保持这一性质。按照构造，这两条规则都只会将变形虫机器人向右推动（从而使其远离穿过最左侧变形虫机器人的 y-轴），且不会将任何变形虫机器人移动到穿过最左侧变形虫机器人的 x-轴下方。◀

现在，我们可以介绍 StarConvex2Monotone 算法，该算法将星形凸结构重配置为 a y-Monotone 结构（不失一般性）。根据定义，我们可以将星形凸结构分解为中心变形虫机器人和至多 6 个互不相交的直方图（见图 16）。我们对每个直方图应用 Histogram2LineSegment 算法，并将与中心变形虫机器人相邻的变形虫机器人作为连接点，从而获得 $2\ y$ 条线段和 $4x$ 条线段。由此得到以下定理。

▶**定理 10.** *StarConvex2Monotone* 算法在 $O(1)$ 轮次内将一个星凸结构重构为一个 MONOTONE 结构。

证明。 注意，根据引理 9，6 个 Histogram2LineSegment 算法实例不会相互碰撞，因为它们始终停留在网格中各自的部分内（见图 16）。运行时间直接由引理 9 得出。

接下来还需证明正确性，即我们得到一个 y-单调结构。得到的 2 条 y-线段与中心变形虫机器人共同构成一条 y-线段。该线段平凡地是 y-单调的。得到的 4 条 x-线段构成两对相邻的 x-线段，并附着于 y-线段。由于这些 x-线段彼此相邻，它们必定是 y-单调的。由于这三个部分都是 y-单调的，该结构是 y-单调的。◀

5 通用重配置

本节中，我们将介绍一种通用重配置算法，它可以将任意结构重构为一条线段。该算法由 3 个子程序组成。在第一个子程序中，我们应用 Arbitrary2Bounded 算法（称为 Arbitrary2Bounded），将该结构重构为有界结构。在第二个子程序中，我们应用一种算法（称为 Bounded2Monotone 算法），将有界结构重构为单调结构。最后，在第三个子程序中，我们应用 Monotone2LineSegment 算法，将单调结构重构为一条线段。最后一个子程序已在第 4.1 节中介绍。接下来，我们将介绍另外两个子程序。

首先，考虑**重构为有界结构**。不失一般性，我们将行数限制为 $\sqrt{n}$。基本思路是反复将少于$\sqrt{n}$ 个变形虫机器人的行与相邻行合并。每次迭代按如下方式进行。首先，我们将一

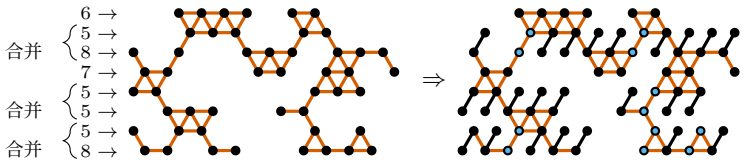
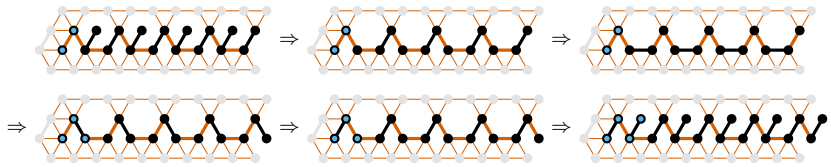
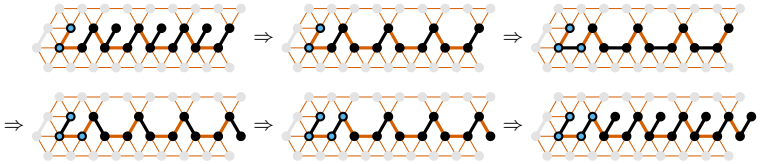


图 17 任意2有界结构算法的第一阶段。变形虫机器人结构有 $n = 49$。因此，我们要合并的每对行中，至少有一行包含少于 $\sqrt{n} = 7$ 个变形虫机器人。蓝色变形虫机器人表示无法扩展的变形虫机器人对。



(a) 偶数个变形虫机器人。



(b) 奇数个变形虫机器人。

图 18 Arbitrary2Bounded 算法第二阶段的单次迭代。蓝色变形虫机器人表示在第一阶段无法扩展的一对变形虫机器人。灰色节点最初与参与的变形虫机器人相邻。

相邻行的最大数量，使得每一对中至少有一行所含变形虫机器人少于$\sqrt{n}$ 个。

然后，我们按如下方式合并每一对行。合并由 3 个阶段组成。在第一阶段，若节点空闲，每个变形虫机器人会平行于 y 轴扩展到另一行（见图 17）。注意，无法扩展的变形虫机器人会形成对（见图 17 中的蓝色变形虫机器人）。

在第二阶段，我们迭代地将这些对中的一个变形虫机器人移动到该行中的下一个空闲位置。不失一般性，考虑最右侧的一对变形虫机器人，以及直到右侧下一个空闲位置之间的所有扩展状态变形虫机器人（见图 18）。其思想是从这对变形虫机器人到下一个空闲位置构造一条交替链。为此，我们将每隔一个变形虫机器人收缩到其底部节点，并确保最右侧的变形虫机器人保持扩展态。该构造保证我们能够维持变形虫机器人结构内的连通性，因为参与的变形虫机器人可以与每个最初的相邻节点至少保持一个键（见图 18 中的灰色节点）。现在，我们可以应用隧穿原语将所有变形虫机器人向右移动（见引理 1）。最后，我们再次扩展所有收缩态amoebot。如果还有另一对变形虫机器人，我们继续下一次迭代。否则，我们进入下一阶段。

在第三阶段，我们再次收缩所有处于扩展态的变形虫机器人（见图 19）。至此，合并完成。由此得到以下引理。

▶**引理 11.** *Arbitrary2Bounded* 算法可在 $O(\sqrt{n}\log n)$ 轮内，将一个任意结构重配置为一个有界结构$(\sqrt{n})$ 结构。

证明。 我们首先证明该算法不会导致任何碰撞。在第一阶段，

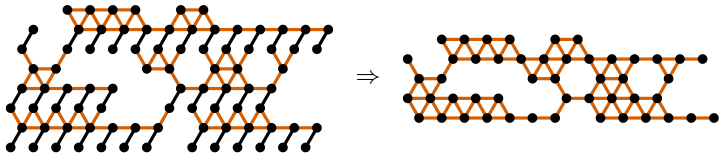


图 19 Arbitrary2Bounded 算法的第三阶段。所有变形虫机器人再次收缩。

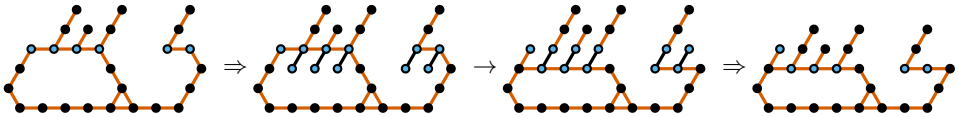


图 20 Aloupis 等人 [4]提出的梳理算法。蓝色变形虫机器人表示扫描线。

变形虫机器人只会扩展到未被占据的节点上。在第二阶段，变形虫机器人只执行孤立交接。在第三阶段，我们收缩整行。因此，这些阶段都不会导致碰撞。由于该算法不会导致任何碰撞，其正确性可由最多只有 $\sqrt{n}$ 行各包含至少 $\sqrt{n}$ 个变形虫机器人的事实得出。

由于我们可以将所有少于 $\sqrt{n}$ 个变形虫机器人的行两两配对（至多有 1 行除外），因此，少于 $\sqrt{n}$ 个变形虫机器人的行数会（近似）减少$_{1\ 2}$。因此，经过$O(\log n)$ 次迭代后，所有行都至少有 $\sqrt{n}$ 个变形虫机器人，从而终止。

每次迭代由三个阶段组成。第一阶段和第三阶段分别只包含单轮。第二阶段的一次迭代需要 5 轮：1 轮用于构造交替链，3 轮用于应用隧穿原语（见引理 1），以及 1 轮用于再次扩展变形虫机器人。

接下来需要界定第二阶段所需的迭代次数。由于每一对行中至少有一行的变形虫机器人少于 $\sqrt{n}$ 个，因此在第一阶段中至多只有 $\sqrt{n}-1$对无法扩展的变形虫机器人对。因此，第二阶段至多只需要 $\sqrt{n}-1$ 次迭代。总体而言，我们需要 $O(\sqrt{n}\log n)$轮次。◀

▶ 注记12。第二阶段可以在每一行内并行化。然而，如果所有变形虫机器人对彼此相邻，我们只能同时隧穿最外侧的两对。在最坏情况下，并行化只能将算法第二阶段的速度提高2倍。因此，运行时间在渐近意义上不会改善。

其次，考虑**重构为单调结构**。不失一般性，我们假设变形虫机器人结构中的行数以 k 为界，目标是将其重构为 y-单调结构。我们采用Aloupis等人 [4]的梳理算法。该算法采用扫描方法。扫描线[3] 从最顶部的行移动到最底部的行。我们移除扫描线上方所有不平行于 y 轴的键，从而得到仅与扫描线相连的 y线段。每当扫描线从一行移动到下一行时，如果线段下方的节点未被占用，我们就将这些线段各自向下移动。为此，我们只需将该线段最底部的变形虫机器人扩展到该节点，随后再将其收缩（见图20）。

▶**引理 13.** *Bounded2Histogram* 算法在 $O(k)$轮内，将 有界结构(k) 重构为 MONOTONE 结构。

[3] 在 [4],中，扫描线被称为墙。

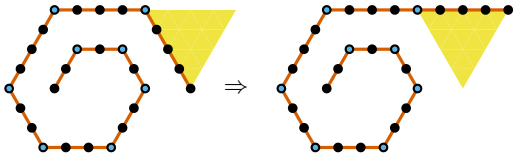


图 21 展开算法。蓝色变形虫机器人表示初始变形虫机器人结构中的拐角。双向箭头表示一个由 2 轮组成的阶段。

证明。 只要能够执行相同的运动原语，我们就可以模拟适用于其他模块化机器人系统模型的任意（重构）算法。因此，正确性和运行时间可由 [4] 推出。◀

最后，通用重配置算法应用 Monotone2Line 算法，将单调结构重构为一条线段（参见第 4.1 节）。结合引理 11 和引理 13 以及定理 8，我们得到以下定理。

▶**定理 14.** 通用重配置算法可在 $O(\sqrt{n}\log n)$ 轮内将任意 结构重构为LineSegment 结构。

6 螺旋

本节介绍两种将螺旋重构为直线的算法。在第 6.1 节中，我们描述一种简单的算法，该算法保留螺旋中变形虫机器人的顺序。在第 6.2 节中，我们展示一种更快的算法，只需常数时间。

6.1 展开

在本节中，我们提出展开算法，将螺旋结构重配置为一条直线，同时保持变形虫机器人的顺序（见图 21）。该算法通过将最外侧线段与下一条线段对齐，迭代地移除拐角。为此，我们只需应用剪切原语。一旦所有拐角均被移除，算法即终止。我们得到如下定理。

▶**定理 15.** 展开算法在 $O(c)$ 轮次内将 Spiral 结构重配置为 LineSegment 结构，同时保持变形虫机器人的顺序，其中 c 表示拐角的数量。

证明。 注意，剪切原语会保持变形虫机器人的顺序。因此，正确性可由归纳法得出，因为每次迭代都会在保持顺序的同时移除一个角。每次迭代仅包含一次剪切原语的应用。根据引理 2，这需要 2 轮次。迭代次数以该数量为界， c，即拐角的数量。总体而言，该算法需要 $O(c)$ 轮次。◀

▶ 备注 16。我们无法通过并行对多个线段应用剪切原语来加速算法，因为最初位于最外侧的线段需要参与的剪切原语数量与拐角数量呈线性关系。此外，并行化可能导致碰撞。

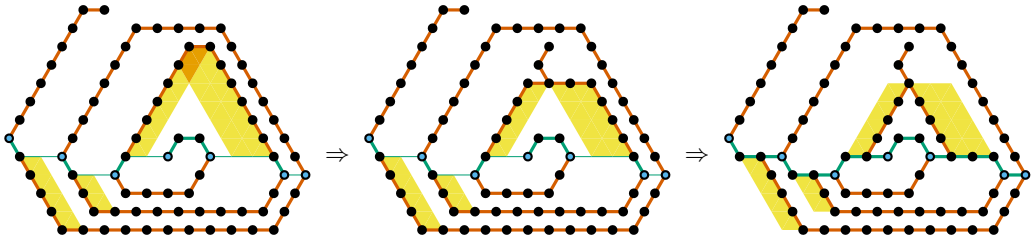


图22 基线构造。蓝色的变形虫机器人表示左、右拐角。绿色路径表示基线。黄色区域表示平行四边形。橙色区域表示平行四边形相交的区域。每个双向箭头表示多个轮次。

6.2 常数时间算法

本节介绍我们的 Spiral2LineSegment 算法。我们假设螺旋至少由 5 条线段组成。否则，我们采用展开算法，在这种情况下，该算法可实现常数时间重构。不失一般性，假设螺旋以一条 y-线段开始，沿顺时针方向向外延伸，并以一条 x-线段结束。我们最多可以执行 2 次展开算法的迭代，以满足最后一个假设。

与通用重配置算法类似，我们沿着一个单调结构进行。因此，算法从 Spiral2Monotone 算法开始，将螺旋重构为 y-单调结构。其思路是穿过螺旋构造一条基线，从而可以切断螺旋的“弧段”而不使结构断开。随后，为了获得单调结构，我们只需将所有不属于基线的线段（不失一般性）与 y-轴对齐。下面将更详细地解释这两个部分。

我们首先如下定义基线 （见图22）。我们从最初的三个最内侧线段开始。从每个左（右）角出发，沿直线向左（右）前进，直到遇到下一条线段（如果存在）。从那里继续前往该线段的下一个左（右）角。如果不存在这样的线段，我们就在该角处结束基线。注意，基线按构造具有 y-和 z-Monotone 性质。

为了构造基线，我们需要填充从左（右）角到左（右）侧下一条线段之间的路径。注意，剩余的基线已经被变形虫机器人占据。我们的想法是并行地对每条遇到的线段及其相邻的 x-线段应用平行四边形原语。但是，平行四边形可能彼此相交，也可能因 x-线段过短而退化。为了解决这些问题，我们可以应用梯形原语。我们选择起点，使其成为平行四边形的一个锐角。

基线使我们能够在每个 x-线段的一个端点处将其断开（最内侧的线段除外，因为它是基线的一部分），而不会使变形虫机器人结构断开。请注意，有些 x-线段可能已缩减为单个变形虫机器人。与三角形原语（见引理 5）类似，我们将所有不属于基线的线段称为“臂”。接下来需要使用剪切原语将这些臂与 y-轴对齐。不过，在此之前，我们必须确保有足够的空间来执行该原语的所有应用。

我们分以下 3 个阶段将臂中的 z-线段对齐（见图 23）。在第一阶段，我们将基线上所有与臂的 z-线段相连的变形虫机器人水平扩展。在第二阶段，我们可以对每个 z-线段应用剪切原语。在第三阶段，我们再次收缩所有变形虫机器人。

接下来，我们分以下 3 个阶段对臂中的 x线段进行对齐（见图 24）。在

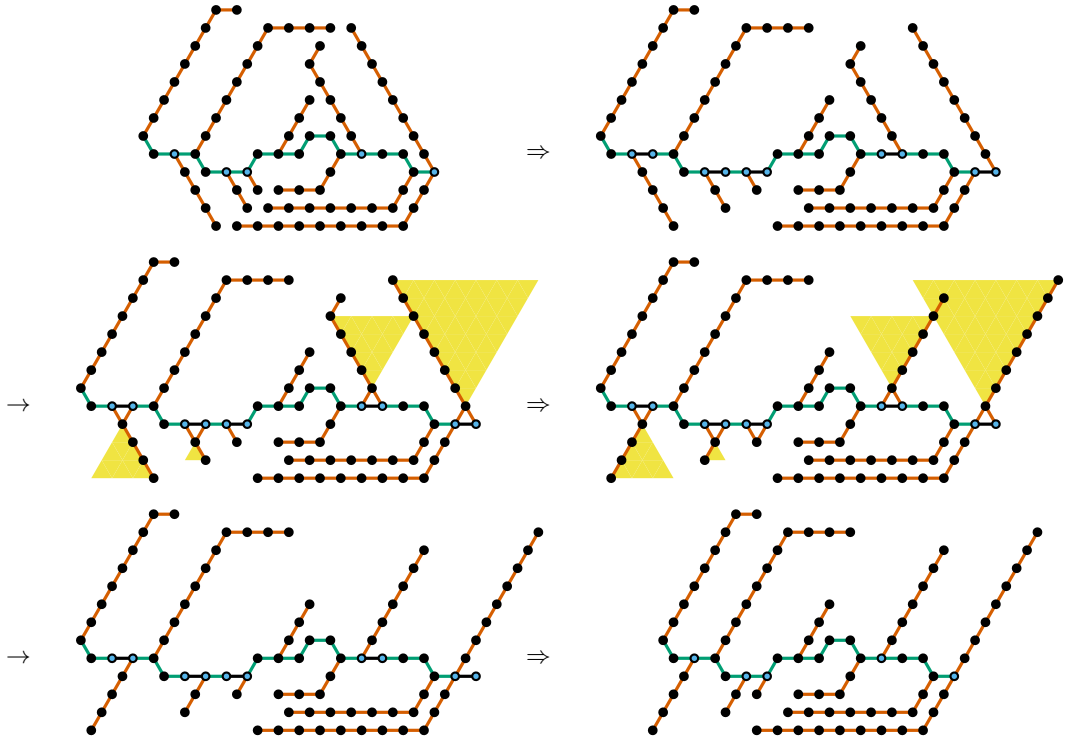


图 23 z线段的对齐。蓝色变形虫机器人表示基线上与某条臂的 z线段相连的变形虫机器人。绿色路径表示基线。黄色区域表示剪切原语。第二个双向箭头表示 2 轮。

第一阶段，我们将基线上平行于 z轴的所有左角和右角，以及臂中平行于 z轴的所有拐角扩展。在第二阶段，我们可以对每个 x线段应用剪切原语。在第三阶段，我们再次收缩所有变形虫机器人。由此得到以下引理。

▶**引理17。** *Spiral2Monotone* 算法可在 $O(1)$ 轮内将SPIRAL 结构重构为MONOTONE 结构。

证明。 根据定义，基线是 y-单调的，并且我们将所有臂对齐到 y轴，因此，只要能证明不会发生碰撞，正确性便可由构造得出。在形成基线的过程中，我们只应用梯形原语和平行四边形原语，这些原语均在原位运行，因此不会发生碰撞。

我们可以将基线分为三部分：由最内侧的三条线段组成的中心部分、由中心部分左侧所有变形虫机器人组成的左侧部分，以及由中心部分右侧所有变形虫机器人组成的右侧部分。此外，我们可以将臂分为四组：

1. 连接到底边基线左侧部分顶部的每条臂由一条 y线段组成，并且可能还包含一条 x线段。**2.** 连接到底边基线左侧部分底部的每条臂由一条 z线段组成，并且可能还包含一条 y线段（由梯形原语产生）。**3.** 连接到底边基线右侧部分底部的每条臂由一条 y线段组成，并且可能还包含一条 x线段。**4.** 连接到底边基线右侧部分顶部的每条臂由一条 z线段组成，并且可能还包含一条 y线段（由梯形原语产生）。

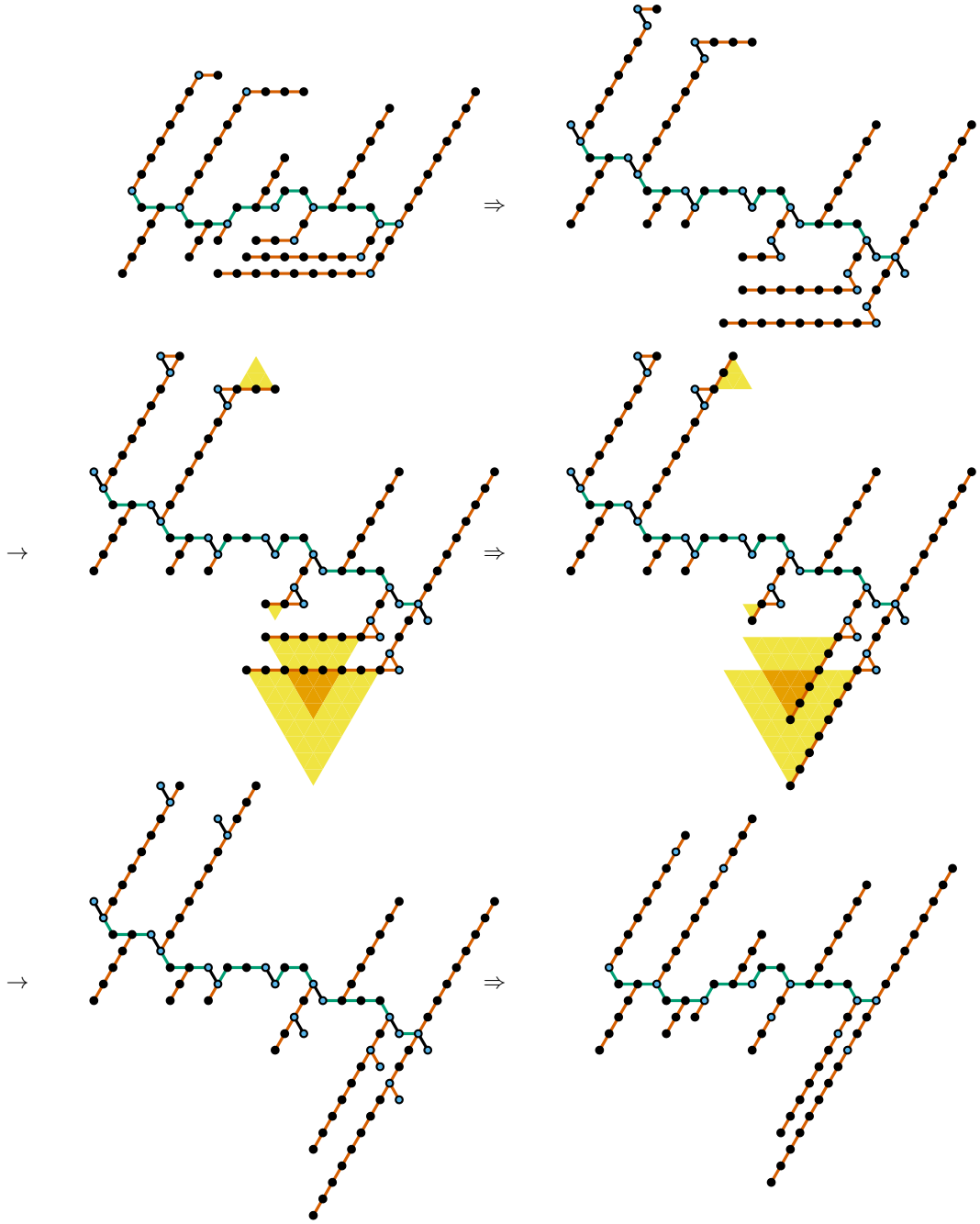


图 24 x-线段的对齐。蓝色变形虫机器人标示基线上的所有左角和右角，以及各臂上的所有拐角。绿色路径表示基线。黄色区域表示剪切原语。第二个双向箭头表示 2 轮次。

考虑臂与基线的连接点。请注意，每个连接点要么是原始的左侧或右侧拐角之一，要么是某个（可能退化的）平行四边形的钝角。这意味着以下性质成立：

1. 对于连接在基线左侧部分底部的每条臂，其连接点左侧相邻的基线变形虫机器人不能位于连接点下方。

24 通过联合移动对可编程物质进行次线性时间重构

2. 对于连接在基线右侧部分顶部的每条臂，其连接点右侧相邻的基线变形虫机器人不能位于连接点上方。

对于其余算法，即 x-线段与 z-线段的对齐，我们需要证明基线和任何组都不会发生内部碰撞，并且没有任何组与另一组或基线发生碰撞。一般而言，若两组位于基线的不同侧，则它们不可能在不与基线本身碰撞的情况下彼此碰撞。因此，只需考虑一个组是否会与基线以及基线同一侧的相邻组发生碰撞即可。

考虑 z-线段对齐的第一阶段。扩展会使各臂彼此分开。由于基线是Monotone的，它不可能发生内部碰撞。由于我们将各臂彼此移开，且臂的形状为两条夹角为 120° 度的线段，各组不可能发生内部碰撞，也不可能彼此碰撞或与基线碰撞。

考虑 z-线段对齐的第二阶段。由于基线和另一组不移动，我们只需论证第二组和第四组臂不会发生内部碰撞，也不会与基线或其他组碰撞。在此阶段，我们对 z-线段应用剪切原语。由于这些线段最初相互平行并以相同速度进行剪切，它们不可能发生内部碰撞。它们也不可能与基线碰撞，因为基线是Monotone的，因此不可能位于剪切区域之内。最后，它们不可能与相邻组碰撞，因为它们是背离该组进行剪切的。

考虑 z-线段对齐的第三阶段。收缩会使我们之前分开的臂重新合拢。由于基线是 Monotone 的，因此不会发生内部碰撞。由于臂要么没有变化，要么以相同的方式变化，因此各组内部不会发生碰撞，组与组之间也不会发生碰撞。此外，由于上述关于连接点的性质，第二组和第四组不会与基线发生碰撞。

考虑 x-线段对齐的第一阶段。暂时只考虑基线上的扩展。这些扩展会使臂分开。由于基线是 Monotone 的，因此不会发生内部碰撞。由于我们将臂彼此分开，而且臂的形状为两条线段构成一个 120° 度角，因此各组内部不会发生碰撞，组与组之间也不会发生碰撞，且不会与基线发生碰撞。现在，考虑臂上的额外扩展。由于我们已经将臂分开，因此有足够的空间将第一组中的 x-线段向上移动，并将第三组中的 x-线段向下移动。

考虑 x-线段对齐的第二阶段。我们只需论证，第一组和第三组臂不会在内部彼此碰撞，也不会与基线或其他组碰撞，因为基线和另一组都不会移动。在这一阶段，我们对 x-线段应用剪切原语。由于这些线段起初彼此平行，并以相同的速度进行剪切，因此它们不会在内部发生碰撞。最后，由于它们会远离相邻组或底边，因此不会与相邻组或基线发生碰撞。

考虑 x-线段对齐的第三阶段。收缩使我们先前分开的臂再次聚拢。由于基线是 Monotone 的，因此其内部不会发生碰撞。由于这些臂要么没有发生变化，要么以相同的方式发生变化，因此各组不会在内部发生碰撞，也不会彼此碰撞。此外，第一组和第三组不会与基线发生碰撞，因为剪切没有改变连接点处的连通性。总而言之，不会发生碰撞，即该算法是正确的。

还需分析运行时间。根据引理 4 和引理 6，形成基线需要 $O(1)$ 轮。根据引理 2，对齐 x-线段和 z-线段分别需要 4 轮（包括基线的建立和恢复）。总体而言，该算法

需要 $O(1)$ 轮。 ◀

结合引理17和定理 8，我们得到以下定理。

▶**定理18**。 *Spiral2LineSegment* 算法可在 O（1） 轮内将Spiral 结构重构为 LineSegment 结构。

7 开放问题

本文展示了如何在次线性时间内将任意结构重构为另一种结构。是否存在多对数时间的通用重配置算法，目前仍是一个开放问题。一种实现此类算法的方法，是设计一种算法，将任意结构重构为有界结构(polylog n)，从而将我们的通用重配置算法的运行时间降至多对数时间。另一种方法是设计一种算法，将任意结构重构为 Rhombical 结构；我们已经知道，如何在对数时间内将其重构为任意其他 Rhombical 结构 [5, 14]。我们还提出了一种算法，可在常数时间内将螺旋结构重构为线段。研究哪些其他类别的结构能够在常数时间内重构为线段，将会很有意思。下一步可以考虑任意路径和树。

此外，我们将分布式算法的设计留待未来工作。回想一下，变形虫机器人之间的局部通信带来了 $\Omega(D)$ 的自然下界，其中 D 是结构的直径。因此，需要采用其他通信方式，例如可重构电路扩展 [11]，它能够实现变形虫机器人之间的快速全局通信。此前的研究表明，可重构电路能够在多种静态问题上实现多对数时间的解 [6, 11, 15, 16]。Almalki 等人 [1] 率先利用它们，在生长模型中实现快速的分布式变换。目前尚不清楚，它们是否也能在采用联合移动的 amoebot模型中发挥作用。

参考文献

1 Nada Almalki、Siddharth Gupta、Othon Michail 和 Andreas Padalkin。利用可重构电路实现形状缩减的高效分布式算法。载于 *SSS*，第 16350 卷，*LectureNotes in Computer Science*，第 40–55 页。Springer，2025。2 Nada Almalki 和 Othon Michail。通过生长操作构造几何形状。理论计算机科学，984:114324，2024。3 Abdullah Almethen、Othon Michail 和 Igor Potapov。推动直线大有裨益：面向可编程物质的高效通用集中式变换。 理论计算机科学，830-831:43–59，2020。4 Greg Aloupis、Sébastien Collette、Mirela Damian、Erik D. Demaine、Robin Y. Flatland、Stefan Langerman、Joseph O'Rourke、Suneeta Ramaswami、Vera Sacristán Adinolfi 和 Stefanie Wuhrer。立方体式模块化机器人的线性重构。 *Comput. Geom.*，42(6-7):652–663，2009。5 Greg Aloupis、Sébastien Collette、Erik D. Demaine、Stefan Langerman、Vera Sacristán Adinolfi 和 Stefanie Wuhrer。使用 $O(\log n)$ 并行移动重构立方体式模块化机器人。载于 *ISAAC*，第 5369 卷，*Lecture Notes in Computer Science*，第 342–353 页。Springer，2008。6 Matthias Artmann、Andreas Padalkin 和 Christian Scheideler。采用可重构电路的 amoebot模型中的形状包含问题。载于 *DISC*，第 356 卷，*LIPIcs*，第 7:1–7:22 页。Schloss Dagstuhl - Leibniz-Zentrum für Informatik，2025。

26 通过联合移动对可编程物质进行次线性时间重构

7 Joshua J. Daymude、Kristian Hinnenthal、Andréa W. Richa 和 Christian Scheideler。可编程粒子计算。载于《移动实体的分布式计算》，第 11340 卷，Lecture Notes in Computer Science，第 615–681 页。Springer，2019。8 Joshua J. Daymude、Andréa W. Richa 和 Christian Scheideler。标准amoebot模型：算法与并发控制。《分布式计算》，36(2):159–192，2023。9 Zahra Derakhshandeh、Shlomi Dolev、Robert Gmyr、Andréa W. Richa、Christian Scheideler 和 Thim Strothmann。简讯：amoebot——一种可编程物质的新模型。载于 SPAA，第 220–222 页。ACM，2014。10 Zahra Derakhshandeh、Robert Gmyr、Thim Strothmann、Rida A. Bazzi、Andréa W. Richa 和 Christian Scheideler。利用自组织可编程物质实现领导者选举与形状形成。载于 DNA，第 9211 卷，Lecture Notes in Computer Science，第 117–132 页。Springer，2015。11 Michael Feldmann、Andreas Padalkin、Christian Scheideler 和 Shlomi Dolev。通过可重构电路协调变形虫机器人。《计算生物学杂志》，29(4):317–343，2022。12 Ferran Hurtado、Enrique Molina、Suneeta Ramaswami 和 Vera Sacristán Adinolfi。二维格基模块化机器人系统的分布式重构。《自主机器人》，38(4):383–413，2015。13 Irina Kostitsyna、Cai Wood 和 Damien Woods。转向机：分子机器人的简单算法模型。《自然计算》，23(2):407–430，2024。14 Andreas Padalkin、Manish Kumar 和 Christian Scheideler。amoebot模型中利用联合移动实现重构与运动。《自主机器人》，49(3):22，2025。15 Andreas Padalkin 和 Christian Scheideler。可编程物质中最短路径森林的多对数时间算法。载于 PODC，第 65–75 页。ACM，2024。16 Andreas Padalkin、Christian Scheideler 和 Daniel Warner。amoebot模型中可重构电路的结构能力。《自然计算》，23(4):603–625，2024。17 Daniela Rus 和 Marsette Vona。晶体机器人：利用可压缩单元模块实现自重构。《自主机器人》，10(1):107–124，2001。18 Tommaso Toffoli 和 Norman Margolus。可编程物质：概念与实现。《国际高速计算杂志》，5(2):155–170，1993。19 Damien Woods、Ho-Lin Chen、Scott Goodfriend、Nadine Dabby、Erik Winfree 和 Peng Yin。多对数时间内算法形状与模式的主动自组装。载于 ITCS，第 353–354 页。ACM，2013。