元胞自动机探究与设计

一、背景介绍

元胞自动机(Cellular Automata, CA)是排列在特定形状网格中的细胞集合体，每个细胞的状态离散，并根据相邻细胞的状态按照约定的转换规则而同时改变。尽管元胞自动机的规则简单，但它却表现出诸多复杂的现象和性质，拥有强大的建模能力，可以模拟交通流、热传导、森林火灾、生命演化等种种复杂的系统。

二、基本原理

（一）一维的元胞自动机

假设细胞的状态只有两种：存活或死亡（分别用 1，0 表示），一维情形只有两个相邻细胞，输入$2^3 = 8$种情形，输出 2 种状态，因此，共计能制定$2^8 = 256$种规则，可由二进制编码顺序进行命名。

以 Rule 30 为例，$30 = 00011110_{(2)}$，对应的细胞存活状态和转换规则如下：

图 2.1 Rule 30

由于一维的元胞自动机随时间演化不够直观，所以我们将演化过程随着时间维度展开成平面，并依次排列。为此，我们可以使用摩尔邻域：第一行即为输入的 3 个细胞状态，所得结果位于第二行中心处，剩余 5 个位置可以视为无细胞存活。则每一行是前一行状态的演化结果。

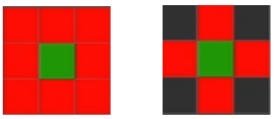

图 2.2 元胞自动机的邻域选择

根据 golly 规则的书写格式，我们只需定义以上 8 条规则即可（其他情形演化过程中也不会涉及）。不难发现：当选用其他规则 Rule N时，只要将最后一列的数字（新细胞的状态）替换为N对应的二进制表示即可。

```
@RULE 30
#二进制表示【00011110】
#复杂无序的分形，混沌

@TABLE

n_states:2
neighborhood:Moore
symmetries:none
0110000010
0100000010
0010000010
0000000011
0110000001
0100000001
0010000001
0000000000
```

图 2.3 Rule 30 规则的实现

对于课堂上实例的手动实现效果如下：（代码在 code 文件夹 1）

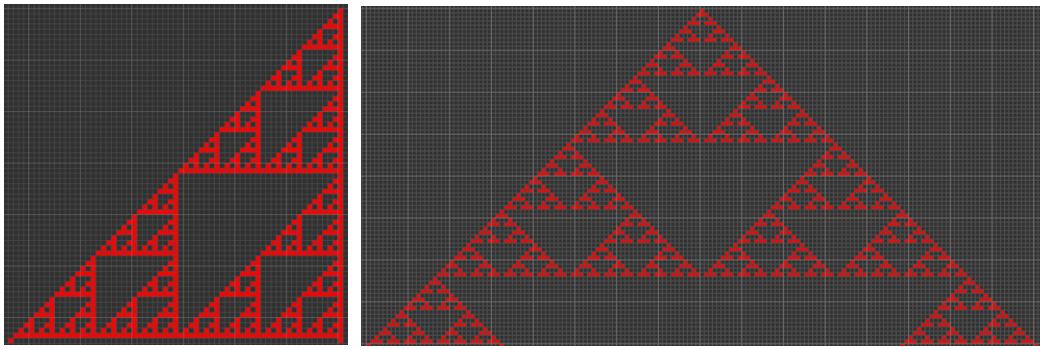

图 2.4 整体有序的规则（Rule 102 和 Rule 22）

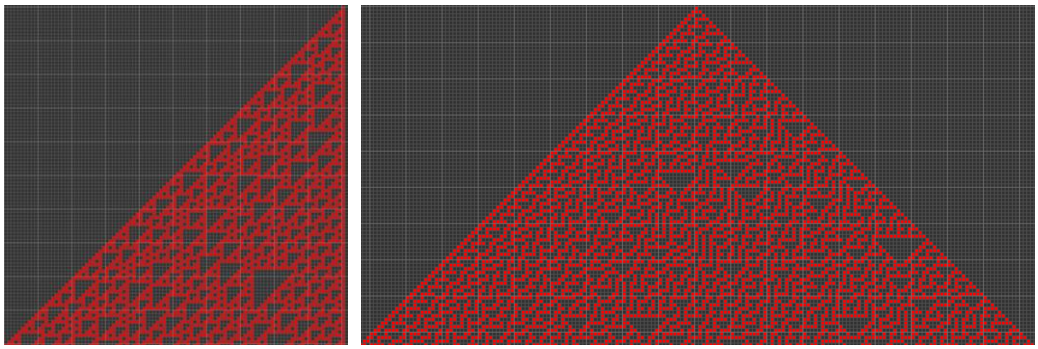

图 2.5 整体无序的规则（Rule 110 和 Rule 30）

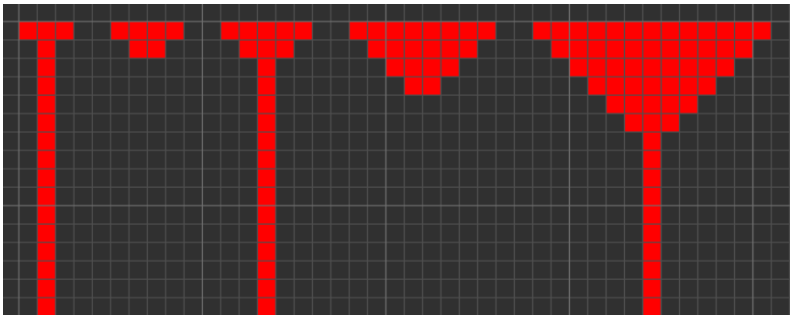

图 2.6 Rule 132 判断奇偶数

（二）生命游戏

生命游戏是在摩尔邻域下的二维元胞自动机，满足规则 B3/S23，即当有且仅有 3 个邻居时，细胞出生；仅在邻居个数为 2 或 3 时，细胞能存活；其他情形，中心细胞死亡。

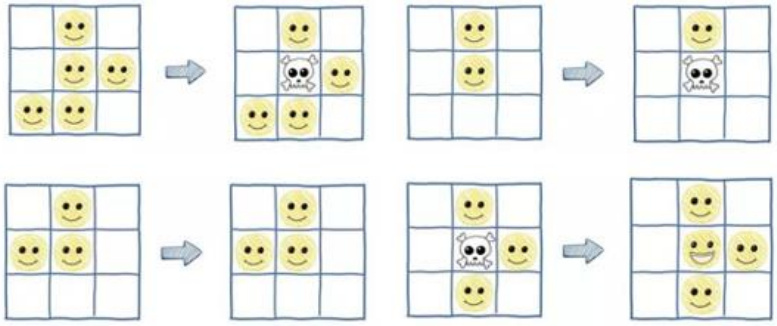

图 2.7 生命游戏规则

在生命游戏中，有许许多多有趣的图案和组件：如飞船、滑翔机、吞噬者、滑翔机枪等，可供我们实现逻辑运算或输出想要的图案，充分利用这些元件就可以实现很多叹为观止的操作：如造一台 8 比特的计算机，包含内存、CPU 等完整组件；在生命游戏中模拟生命游戏；制作更高分辨率的生命游戏显示器等等。

我们考虑使用滑翔机环、飞船实现“BUAA 70”的字样输出。

三、实验设计

1. 熟悉基本组件

飞船和滑翔机的周期为 4，用于传播信号和输出图案；双蜂梭（Twin bees shuttle）周期为 46，可以使滑翔机直角转弯。其示例图如下：

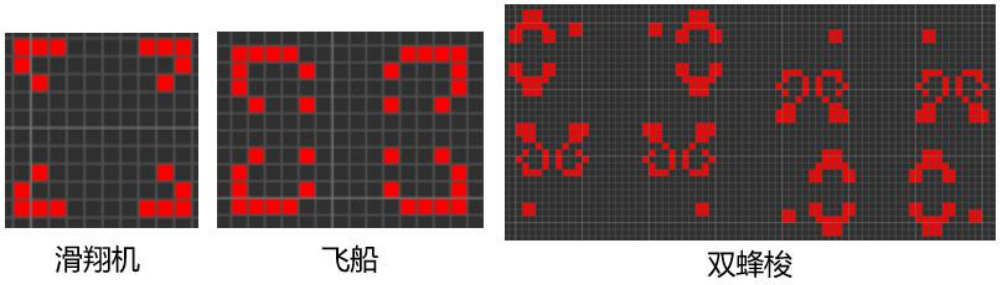

图 3.1 部分基本组件的示意图

2. 模型设计架构

要源源不断地输出特定序列的字符，我们需要对序列进行存储。已知如下结构可以将滑翔机序列进行复制，得到两组平行且同步的输出。将其中一组作为输出序列，另一组进入一个闭合回路，继续存储并重复用于输出即可满足要求。

又因为环形结构的限制，各组序列不能直接相互平行地输出（会相互干扰），所以我们将滑翔机运动方向进行 135° 的旋转，转换为飞船，作为最终输出。基于上述分析，我们得到传送带状的“回路”结构。

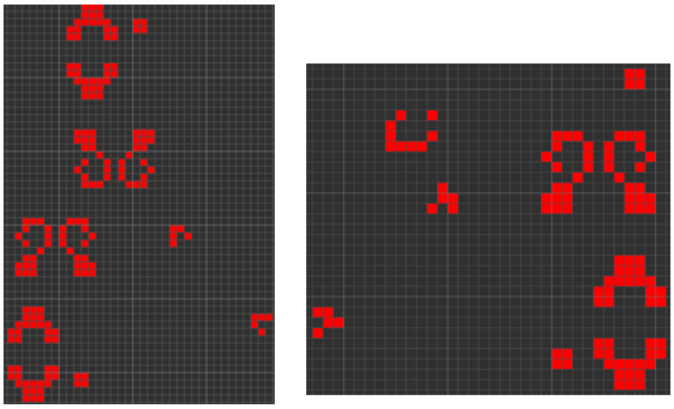

图 3.2 复制和转换滑翔机的组件

3. 获取点阵表示

在文本框中敲出“BUAA 70”的字样，截图后粘贴到 golly 中即可获得字符的点阵表示，稍加调整即可，并确定出总行数和一个循环的总长度。总行数即为传送带“回路”的个数，循环的总长度决定了“回路”中滑翔机的个数。

由于每个传送带之间相差 6 个飞船，所以我们将“BUAA 70”的字符依次错位 6 个单位，去掉一个完整周期间的固有间隔，重新拼接起来。按照序列给每个传送带中的滑翔机组做删减即可。最终输出如图：（具体组件和实现在 code 文件夹 2）

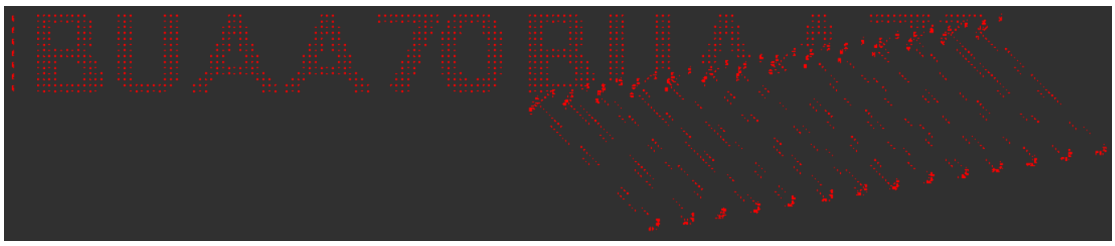

图 3.2 北航 70 周年的输出字样

四、总结与结论

虽然实验中遇到了不少困难和挫折，因为 CA 的演化过程正向推演远远易于反向推导（甚至很多细胞的排列和组合情况不能由某一状态演化而来，即不存在原状态），在实现时“失之毫厘、差之千里”，出现了“滑翔机间存在相位差”，“回路出现偏差、无法闭合”等问题，但所幸都一一克服了。总的来说，我的收获颇丰：对元胞自动机的原理及其应用有了具体全面的了解；熟悉并掌握了 golly 的编写规则和使用方法（比 python 和 C 语言在实现元胞自动机算法上更直观、更高效）。

一维的元胞自动机仅考虑两种生存状态、3 个细胞，其中的 Rule 110 就已经图灵完备了，可以实现复杂的逻辑运算。如果我们将细胞所处空间的维度升高、邻居范围增大、用细胞的个数来表征状态，并相应地赋予演化规则，其复杂度将大大提高。由此可见，“规则简单”的元胞自动机却具有远非我们能理解的“复杂”。