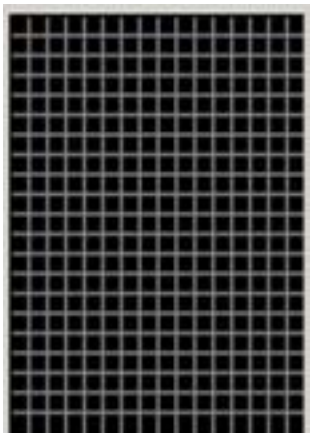
方格宇宙

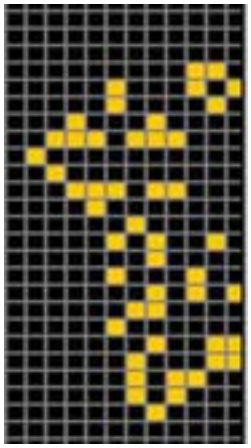
1、大楼霓虹灯动画

你看过夜晚都市街头的霓虹灯么？它们是多么变幻莫测、色彩缤纷！然而你想没想过，这些霓虹灯的动画图案都是由很多单个的灯泡不停闪烁构成的，你不觉得奇怪么？让我们定睛观察其中的一个灯泡，它只会机械的闪烁变换几种不同的颜色，没有什么稀奇的；然而当你观察由这些小灯泡构成的集体的时候，霓虹灯整体的动画图案就会像变魔术一样突然表现出来。这意味着什么？不忙回答这个问题，让我们先来玩一个类似霓虹灯的游戏。

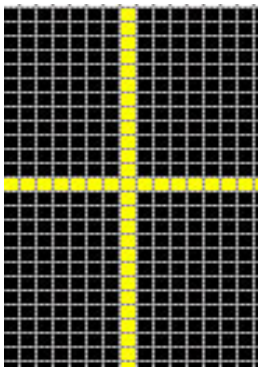
设想你和你的朋友居住在一幢大楼里面，大楼的一个侧面有很多的窗户，他们整齐的排列成一个大方阵，就像下面的图一样：



这里黑色表示的是窗户，灰色是窗户之间的隔断。为什么窗户是黑色的？太难看了吧？哦，这是晚上，而且屋子里都没有开灯。没关系，反正大楼里面住的都是自己的朋友，你可以让一些人把灯打开，下面的图表示的是有些屋子亮灯的图案（好看多了）：



聪明的人已经想到了，其实这就是一个简单的霓虹灯！我们可以让不同的窗户开灯来组成不同的图案。比如要想用这种方法组合出一个“十”字图案，我们可以让第 7 列的窗户都开灯，并让第 13 行的窗户也都开灯：



如果是复杂点的图案呢？我们仍然可以如法炮制，不过麻烦了点，因为你需要告诉这幢大楼里的每一个同学是开灯还是关灯。

如果要让霓虹灯图案动起来呢？比如每 1 秒钟就变换一下图案，那么你就需要给每个朋友编制一些指令，告诉他们在什么时刻是开灯还是关灯。我们不妨用 0 表示关灯，用 1 表示开灯，那么比如指令 011011001 指派给了住在第三行第 4 列的朋友，那么他应该在第 1 秒内关灯，在第二秒、第三秒都开灯，第四秒关….。汗，这恐怕太麻烦了，你算算，如果要展示一个 10 分钟的动画（600 秒）那就需要为每个窗户后面的朋友编制长度 600 的指令。何况很可能这里有 100 扇窗户呢！

如果你够聪明，可以换一种方法来制作你的大楼霓虹灯动画。让居住在大楼里面的每个朋友密切观察他的邻居，根据上一秒内他邻居的开灯状况决定下一秒内该朋友的动作。显然每个窗户后的朋友都会有上、下、左、右、左上、右上、左下、右下八个邻居（如下图），那么每个人的开灯状态就会被他的邻居们以及他自己的上一时刻的开灯状态所影响，并且翻过来这位朋友是否开灯又会影响他的邻居们，整个大楼的各个窗户就会不停的闪烁，这也能达到动画的效果了。

一个窗户和它的八个邻居，分别是左上、上、右上、左、右、左下、右下

具体邻居的开灯与否如何影响该朋友是否开灯呢？这仍然需要我们为该朋友定义规则，比如“只有当八个邻居的灯都亮了我才能开灯”，这就是一条规则，“只要我这个时刻亮着灯，那么下一时刻无论邻居们什么情况我仍然亮灯”也是一条规则……。另外，还有一个初始状态的问题，也就是在一开始第一秒内哪些窗户应该开灯，那些应该关灯呢？我们完全可以让窗户后面的每个朋友自己随便决定，也就是初始时刻窗户的开灯状态是随机的。

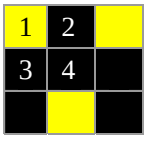
2、生命游戏

好了，上面基本讲清楚了大楼霓虹灯的工作原理了，下面我们开始设计一个切实的动画。按照刚才讲到的聪明的霓虹灯动画设计方法，我们为每个窗户后面的朋友设计规则如下：

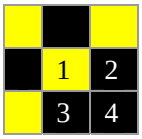
规则 1：如果我现在开着灯，只要八个邻居中有 2 个或者 3 个人开灯了，那么下一时刻我才开灯，否则邻居亮灯数超过 3 个或者少于 2 个，我都会把灯关掉；

规则 2：如果我现在关着灯，那么当邻居中有 3 个人开灯，那么下一个时刻我就把灯打开。

比如某个窗户和他邻居窗户的开灯状态如下图示：

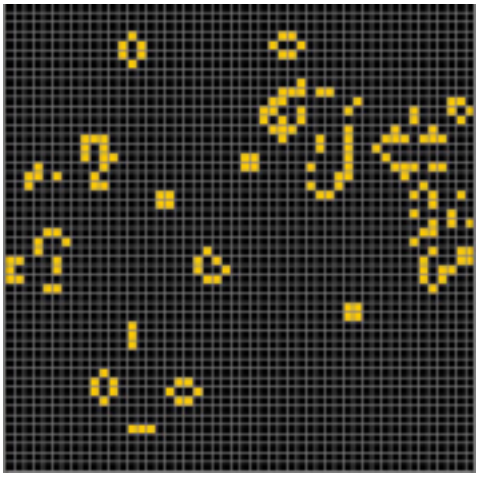


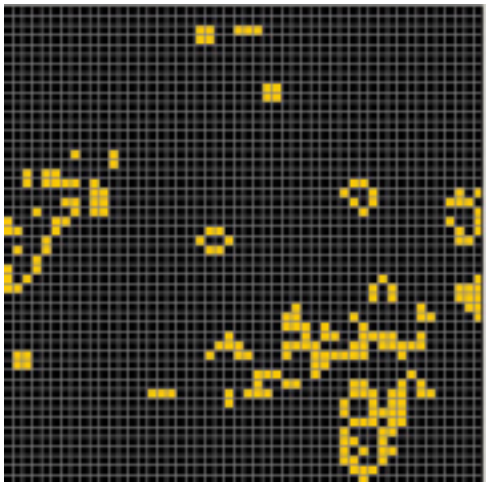
其中中心位置的窗户为你所在的窗户（也就是图中的 4 号），在这个时刻，你观察旁边的八个邻居中有左上、右上、下三个窗户是亮灯的，那么按照规则 2，你在下一个时刻应该把灯打开。这样，每个窗户后面的朋友都按照这两条规则行事，比如左上方 1 号这个邻居下一个时刻是否应该开灯呢？那么我们就应该考察这个窗户的 8 个邻居，以 1 号窗户为中心的 9 个方格如下图：



那么根据规则 1，下个时刻应该继续开着灯。注意，这个图中的右下部分标有 1、2、3、4 的 4 个窗户与上图中左上角标有 1、2、3、4 的 4 个窗户的开灯状态是一样的。他们都是同样的 4 个窗户。

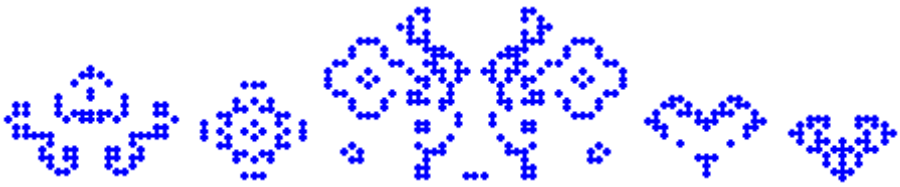
这样，当我们用同样的规则考虑大楼中所有的窗户的时候，整体大楼的动画就可以产生了。我们以一个随机的初始状态开始，根据那两条规则会产生什么样的大楼霓虹灯动画呢？让我们把镜头拉远，观察整个大楼：哇，一个奇妙的动画诞生了。下面是两张动画的截图（由 50 行 50 列窗户构成的大楼）：



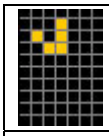
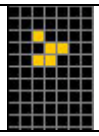
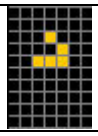
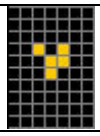
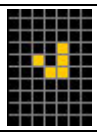


很怪异不是么？如果你看到能够运动的动画你会感觉更怪异，整个大楼的窗户就像沸腾了。实际上，我们可以把整个大楼霓虹灯游戏做在电脑里，让电脑程序代替那些居住在大楼里，不停机械的开灯关灯来完成你设计的无聊动画的可怜的朋友们。你可以到以下网址观看程序：gameoflife/。在程序中，方格就代表窗户，黄色的格子就是亮灯的窗户，黑色的就是关灯的。

你会发现，整个电脑屏幕像煮开了一样，不停的有方格变成黄色，也有方格变成黑色，而由这些亮方格构成的图案是五花八门，奇形怪状，看这是几个截图：



这些图案有风筝、蝴蝶、桃心。如果观察动态结果，你会发现不可思议的东西还更多。首先，你会发现一种被人们称作“滑翔机”的结构，他是由几个格子构成的一个动态的结构，随着时间的推移，他会从屏幕的一段走到另一端，下面是“滑翔机”行走的原理：

				
1	2	3	4	5

在 5 步内，“滑翔机”结构往右下方行走了一个格子。你可以看到，在第五步内黄色亮点组成的结构与第一步的结构完全一样，只不过位置完全平移了。“滑翔机”会偶尔在屏幕的一个角落里闪现出来，步态缓慢的移动到另一角，就好像一个悠闲的散步者，在楼宇的窗户间移动着。值得注意的是，你可以推敲一下，“滑翔机”的每一步移动都完全符合我们上面定义的两条动画规则。

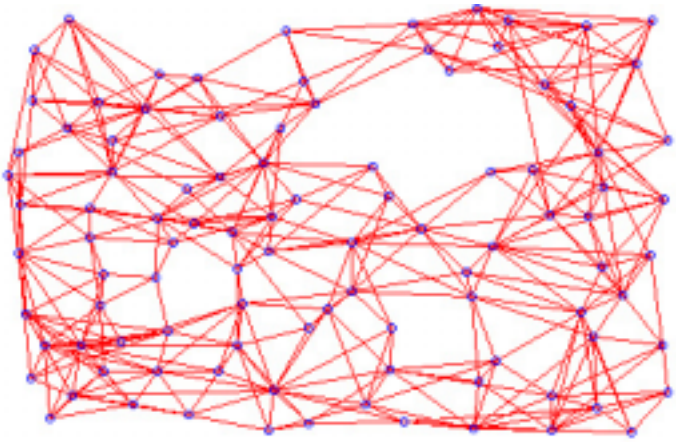
我们的游戏还有其他的壮观景象：看这边由多个亮窗子构成的两个“村落”分处于大楼的角落彼此没有来往，然而一个“村落”发射了一个“滑翔机”对另一个“村落”展开了进攻，而另一个“村落”则由于这个“滑翔机”的到来被打破了原先的寂静开始沸腾起来。偶尔你会看到一个村落逐渐变大、分化成两个部分分别进行繁衍……，这里的语言不能穷尽这个游戏的所有可能情况。

该游戏有一个好听的别名：“生命游戏”，这是由计算机科学家 John conway 在上个世纪 50 年代发明的。为什么叫做“生命游戏”呢？其实你不妨大胆运用你的想象力，认为这个游戏创造了一个“活生生的”宇宙。也就是说你和你的朋友们遵循简单的规则产生的集体楼宇霓虹灯动画创造了一个宇宙，这个宇宙就镶嵌在你们的楼宇的窗户之间。该宇宙按照自己的规律不断演化着。我们可以把闪亮的格子看作是这个虚拟宇宙中的生命体，这样，它们不停的演化、繁殖、灭亡。

3、从霓虹灯到方格宇宙

这一切又意味着什么呢？从霓虹灯的例子中我们体会到了系统的深层含义。什么是系统？由多个部分组成的整体就是一个系统，上面提到的霓虹灯的例子就是一个系统，它是由多个小灯泡的组合形成的一个霓虹灯整体；人类社会也是一个系统，各种各样的人构成了这个系统；“生命游戏”也是一个系统，多个方格格子组成了它。仅仅了解系统的部分并不能了解整个系统。这就好像你能很清楚的了解到每个单独灯泡的闪烁状态并不能帮助你理解多个小灯泡构成的霓虹灯图案如何的道理一样。要想知道霓虹灯整体的图案，你必须要“跳出来”，也就是站在一个更高的角度来观察这个霓虹灯的整体，而忽略每一个具体灯泡的细节。我们常说的只见树木不见森林正是揭露了观察系统的时候太重视局部而忽略整体的这个弊端。

我们学到的科学方法一直在运用分解、还原的方法。比如要研究一台电视机，我们就需要把这台电视机拆开，看看每一个部件的工作机理，这显然是有效的，但是正如你仅仅有一大堆电子元件而不会正确的组合它们就不能组装一台电视机一样，单纯运用分解还原的老方法是不能够理解系统。这又是为什么呢？原因是系统的特性一方面体现在组成部分的性质上，而更多的则取决于系统各个部分的相互之间的关系上。正是由于系统中各部分的相互关系的复杂性才导致了系统的复杂性和不可理解性。观察下面的图：



我们可以认为多个蓝色的点构成了一个系统，而每个点之间的连线就是系统组成部分之间的联系。为什么系统很难理解呢？原因是点之间的连线太复杂了。假如一个系统由 n 个点组成，那么所有可能的点之间的连线关系就有 $2^{(n-1)(n-2)/2}$ 种情况，比如 n=100，那么连线的可能性就是 2^{4851}，不用算了，这个数字已经远远超过了当前的宇宙年龄。我们之所以对系统概念感到很难理解正是由于系统组成部分之间关系的复杂性。而当我们用还原的方法研究系统的时候，也就是要研究系统中单个点的性质，我们就必须切断该点与其他点的联系，这当然方便我们研究该点，但是当切断它与其它点的联系的时候，我们已经严重破坏了系统的性质，看不到了系统各部分之间的联系了。在霓虹灯的例子中，当我们把目光集中到单个灯泡上的时候，我们就是采用还原分解的方法把注意力集中到了一个点，而没有看到该灯泡与其它灯泡之间的联系，所以我们不能够清楚整体霓虹灯的图案与动画。

好了，到这里如果你已经完全理解或者隐约理解了上面所讲的系统概念，那么你已经取得了一个重大的进步。下面我们再来看看系统的另外一个性质。我们都知道计算机是由硬件和软件组成的。什么是硬件什么又是软件呢？运用我们刚刚熟悉的系统的概念不难回答，广义上来讲构成系统的各个单点就是硬件，而点与点之间的连线关系以及这些关系的变化就是软件。比如一台计算机系统的硬件部分是由 CPU、内存、硬盘、显示器、键盘鼠标等等部分组成，那么软件就是运用这些硬件的一种组合方法。比如 Word 程序这个软件就是把键盘输入的信息经过 CPU 和内存的处理显示到显示器上，这整个过程就是一个软件。当然我们这里的叙述忽略了很多技术细节，而仅仅在概念形式上进行讨论。我们知道硬件构成了软件的基础，显然你没有硬盘、内存、CPU 这些硬件是不可能执行任何软件程序的，但反过来，软件又是独立于硬件的某种特有的东西。因此你在一台 PIII 的机器上运行 Word 程序和在另一台 PII 上运行该程序一样（除了速度不同）。

知道了硬件与软件之间的关系有什么用处呢？它可以帮助我们解答关于生命、智能、社会等等难以理解的复杂事物。比如上面介绍的“生命游戏”就是一个很好的例子。因为整个游戏的过程就是一个软件，这个软件无论运行在何种硬件介质中都是无关紧要的。所以我们用大楼窗户的开灯与关灯可以完成“生命游戏”，而运用电脑屏幕中的闪烁的方格也可以来完成该游戏。显然，我们现实的生命过程要远远比生命游戏复杂得多，然而它也很有可能是一种软件的过程，而与运行该软件的硬件没有关系。同样的，人类的智能也是一种复杂的软件过程，用人类的头脑能够完成的思维智能过程也同样能在另一种介质：计算机中实现。这也正是“人工智能”的理论来源。

进一步，我们可以大胆设想：把整个宇宙都理解为一种运算的软件过程，这样，完成该运算过程的硬件介质是无关紧要的，而关键是软件过程。在“生命游戏”中，我们正是创造了一种宇宙演化的过程。在游戏中，不同的方格可以理解为抽象的宇宙空间，而程序运行的嘀嗒声正是这个虚拟宇宙中的时间。

在“生命游戏”中，我们规定了每个格子的运行规则而得出了一个虚拟的宇宙，如果我们设计其他不同的规则会得到什么样的宇宙呢？进一步的，在大楼霓虹灯的比喻中，如果我们允许每个窗户里面灯的颜色有多种，每个窗户的邻居窗户不仅仅是 8 个，而可以是 24 个或者 48 个能演化出什么样的宇宙呢？更进一步设想，如果我们运行的游戏是在一个三维空间中或者四维空间中又是什么样的呢？所有这一系列的问题几乎可以写成一本教科书了，当然这本教科书中并不教我们牛顿定律，而是一套关于我们自己的虚拟宇宙的物理学知识。我们为什么一定要学习这个宇宙的科学？难道我们不能自己作上帝创造一个全新的宇宙么？嗯，我们就把我们的宇宙叫做“方格宇宙”吧，下面就来看看这个“方格宇宙”有什么性质。

4、细胞自动机

下面的篇幅我们将具体讨论我们的“方格宇宙”的细节性质和规律。实际上，“方格宇宙”有一个学名叫作“细胞自动机”或者“元胞自动机”，英文名字是“Cellular Automata”，下面的讨论中，“方格宇宙”和“细胞自动机”这两个名字我将混合使用，你要知道这两个指同一个东西就可以了。

一般的，一个细胞自动机是由这几个要素构成的：细胞、细胞空间、邻居、规则。下面分别叙述。

（1）细胞

在细胞自动机中，一个小格子就是一个细胞。在大楼霓虹灯动画中的窗户就是一个细胞。细胞具有状态，比如大楼霓虹灯中每个窗户的开灯还是关灯就是细胞的状态。我们把每一个细胞的所有可能状态的集合叫做细胞的状态集。在大楼霓虹灯中每个窗户有开灯关灯两个状态，我们可以用集合的符号表示为{开灯，关灯}，因此这是一个两状态的集合。更一般的，一个细胞可能有 n 种状态。

每一个状态都可以给他赋予不同的含义，比如在“生命游戏”中，我们可以用状态集{0，1}表示一个细胞的状态。也可以用{生,死}表示一个细胞（生命体）的状态，当然也可以用{开灯，关灯}表示运行在大楼霓虹灯上的“生命游戏”这个软件。对于有 n 个状态的细胞自动机，我们不妨把这 n 种状态想象成 n 种不同的颜色，这毫不影响我们对 n 状态细胞自动机的理解。仍然用大楼霓虹灯做比喻，如果每个窗户后面的灯是有颜色的，比如有红、蓝两种颜色，那么每个窗户就有{红色灯，蓝色灯，关灯}三种状态。当然我们也可以把关灯理解为黑颜色灯的状态，所以这和用{红灯，蓝灯，黑灯}的表示是一样的。

（2）细胞空间

简单地说细胞自动机中细胞分布的空间就是细胞空间。在“生命游戏”中，细胞空间就是一个二维的大方块，并且所有的细胞都是以水平和竖直两个方向整齐排列的。关于细胞空间有以下几个概念：

a、维度：

也就是细胞空间的维度。常见的细胞空间都是一维或者二维的，一维的细胞空间就是在一条直线上的方格，如下图：

二维的细胞空间就是一个平面，我们前面介绍的“生命游戏”就是一个二维的细胞空间。三维的细胞空间是一个立体的立方体，下面给出了示意：

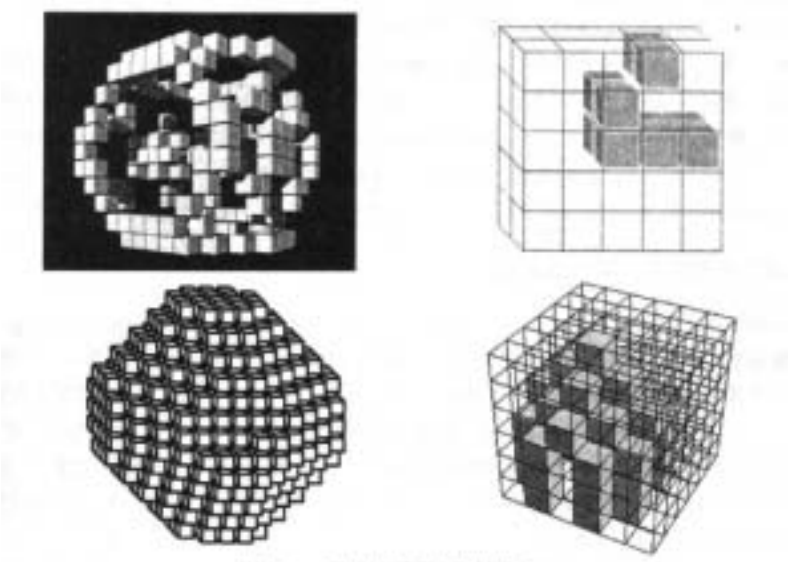


图 2-8　三维的元胞自动机模型

当然我们还可以给出 4 维细胞空间、5 维细胞空间等等，但那已经远远超出了图形表达的能力极限了。

B、细胞空间的几何划分：

细胞的形状以及细胞之间的排列方式。我们常见的是横平竖直的四方格排列方式，当然还可以有其他的几何划分，如下面的几个例子：

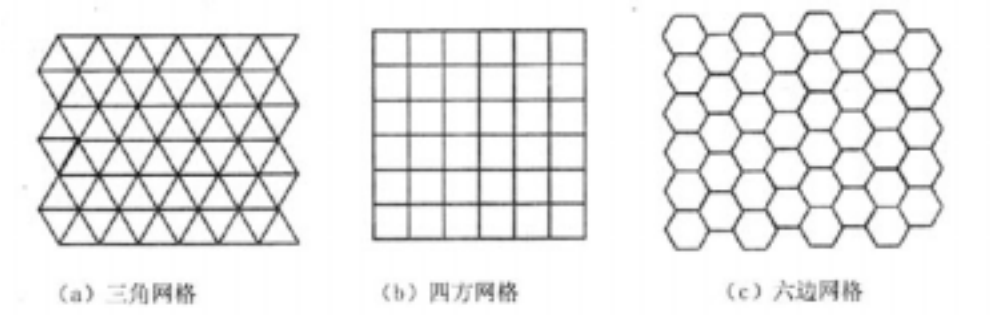


图 2-5　二维元胞自动机的三种网格划分（根据李才伟，1997）

这些几何划分分别是三角构形，四方构形和四方网格还有六边形网格。一般的，只要某种图形能够形成对一个平面区域的覆盖就可以认为是一种细胞空间的几何划分。

C、边界处理方法：

细胞空间边界处细胞的处理方法。无论是哪一种细胞空间，为了能在计算机上进行模拟它都必须是一个有限的区域，因此我们必须考虑在边界上的细胞空间处理方法。比如在“生命游戏”中，它的空间是一个方块，那么在方块四个边上的细胞应该如何处理呢？要知道左边的方格没有更左边的邻居了。一种很自然的想法就是把最右边边界处的细胞当作是这个细胞的左邻居，而右边边界处的那个细胞的右邻居是该细胞。也就是说细胞空间的左右两端被连接在了一起，就好像是一个圆柱面。同样的方法也可以应用到上下两个边界上。

在细胞自动机的实际应用中，还可以用反射型、固定值型的方法处理边界这里不再多说了。

（3）邻居

细胞自动机的运行规律不仅仅取决于某一个细胞自身的状态，而且也取决于它周围的多个细胞的状态，这些细胞就是该细胞的邻居。我们可以采用多种方法定义一个细胞的邻居。

在方格细胞空间中，常用的邻居类型包括下面几种：

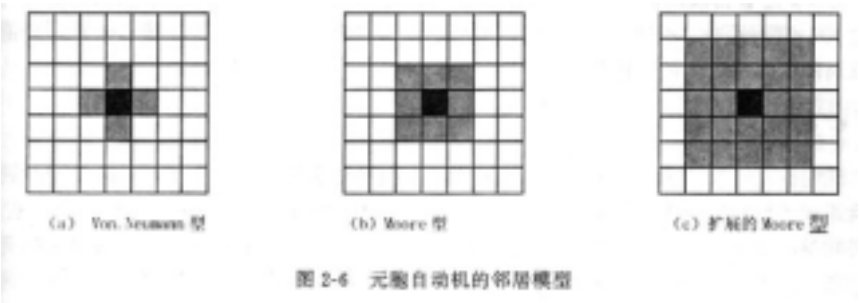


图 2-6 元胞自动机的邻居模型

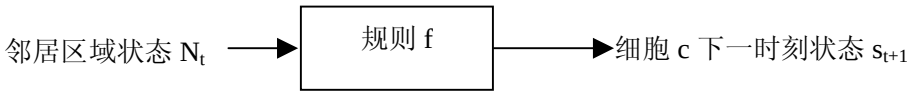
这其中，黑色的方格表示当前的细胞，灰色的表示它的邻居。上面的三种类型都是人们常用的类型。在实际应用中，还有很多邻居的类型。另外，邻居的类型也跟细胞空间的几何划分和维度有关系。在一维的细胞自动机中，我们常常用一个细胞在一个方向相邻的细胞个数定为该细胞自动机的邻居半径，比如邻居半径为 1 的一维细胞自动机的邻居区域如下图示

邻居半径为 2 的一维细胞自动机的邻居区域如下图示：

一个细胞和它的邻居一起的空间区域被叫做**邻居空间**或者邻居模板。

（4）规则

正如在第一节中叙述的，要想产生动态的过程，每个细胞都需要遵循一些规则改变自身的状态。这些规则就好比是一些指令指导着每个细胞根据它自身的状态和它周围邻居的状态不断的改变自身。我们也可以把每一条规则看成是一个机器 f。它的作用是把当前方格 t 时刻的状态和他的邻居在 t 时刻的状态转换成 t+1 时刻该方格的状态。如果用 s_{t+1} 表示在 t+1 时刻该细胞的状态，而用 N_t 表示 t 时刻细胞邻居区域内所有细胞的状态（包括这个细胞自身在 t 时刻的状态），那么利用规则函数 f，我们可以得到下面的关系式：$s_{t+1}=f(N_t)$。或者，我们把细胞自动机的规则看成是一个魔术箱子，输入给它的是邻居区域所有细胞的状态 N_t，输出的是这个细胞的下一时刻的状态 s_{t+1}。用下图表示：



比如“生命游戏”中，某个细胞核它的八个邻居在某一时刻 t 分别处于状态 0（自己）,1（左上）,1（上）,0（右上）,1（右）,0（右下）,0（下）,0（左下）,0（左）。这样 N_t 可以看作是一列：(0,1,1,0,1,0,0,0,0)，根据“生命游戏”的规则 2，那么在 t+1 的状态就应该是 $s_{t+1}=1$。也就是说 $s_{t+1}=f(N_t)=f(0,1,1,0,1,0,0,0,0)=1$。

由所有的规则组成的集合我们就叫做规则集。比如在生命游戏中，规则 1 和规则 2 构成的集合就是一个规则集。这里可能会有一个问题，万一某一种输入状态找不到对应的规则怎么办呢？比如在我们的生命游戏的规则集合中，如果仅仅有规则 1，而没有规则 2。这样当某个窗户后的朋友关灯的情况下就不知道下一时刻到底是开灯还是关灯了。因为规则 1 仅仅定义了如果当前窗户是开灯的情况下该如何转移到下一个状态，而对于当前窗户关灯的情况没有定义。显然，如果出现这种情况我们的游戏就不能玩下去了。因此，关于规则的集合我们必须有下面的要求：规则集要能对所有可能的输入状态组合（当前邻居区域中的细胞状态的所有可能组合）都有相应的处理。

到这里，我们已经介绍了一个一般的细胞自动机的基本要素，可以说这些基本要素在不同的情况下会造就完全不同的“方格宇宙”。接下来的问题自然是，由这些要素到底能得到

什么样的“方格宇宙”呢？以及得到的这些“方格宇宙”会有什么样的“物理规律”呢？我们下面就从一维的细胞自动机开始探索“方格宇宙”的“物理规律”。

5、解刨“方格宇宙”

（1）最简“方格宇宙”

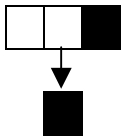
为了讨论简单起见，我们先对最简单的“方格宇宙”进行解刨。最简单的“方格宇宙”当然是 1 维的细胞空间，状态集合为两个元素{0,1}。邻居是一个半径为 1 的区域，也就是每一个方格的左、右两个方格是它的邻居，这样每一个方格单元和它的邻居可以表示如下：

黑色的方格是当前细胞，两边的灰色方格是它的邻居。由于状态集只有{0，1}两个状态，也就是说方格只能有黑、白两种颜色，那么任意的一个方格加上它的两个邻居，这 3 个方格的状态组合一共就有 8 种。这 8 种情况为下图示：

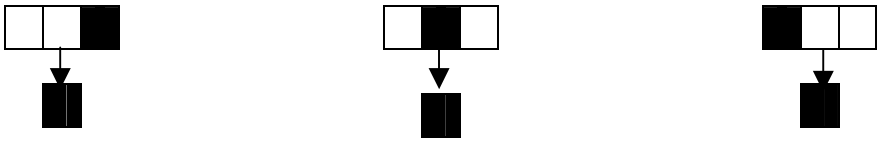
他们表示的状态分别是：000，001，010，011，100，101，110，111。也就是说对于状态数为 2，邻居半径为 1 的所有一维细胞自动机的邻居和其自身的状态组合就这 8 种。

（2）规则与规则集*

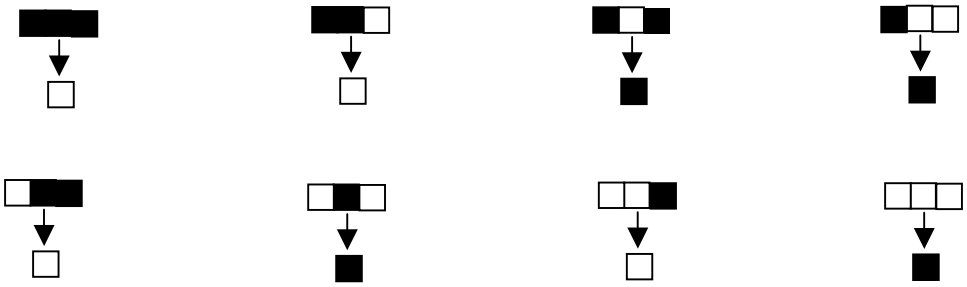
下面考虑规则。前面已经说了，细胞自动机的规则就是根据每个细胞和它的邻居的当前状态转移到下一个时刻该细胞的状态。由于在我们这个最简单的细胞自动机中每个细胞和它的邻居状态的所有可能组合就上面列出来的 8 种，所以无论规则是什么样的黑箱，它的输入就是上面列出的 8 种组合之一，因为表示的是每个细胞下一时刻的状态，而状态只可能有 0、1 两种，则规则的输出要么是 0，要么是 1。这样，任何一个规则就是一个或者一组转换，比如下图表示的就是一条规则：



另外，若有一个规则是：“如果输入的三个方格中黑色方格只有 1 个，那么下一时刻当前方格就是黑色”可以表示成下面的图：



下面我们再把目光转到规则集上。由于每一条规则都是一个状态或一组状态的转换，那么规则集也就是一组状态转换，把输入的 8 种可能情况转换到当前细胞的下一状态。我们可以用一个转换表表示一组规则集，例如：



这个规则集也可以用下面的一组数字表示为：

输入	111	110	101	100	011	010	001	000
输出	0	0	1	1	0	1	0	1

每一组规则集也可以表示成类似于上面的图和表，例如下面的另外一组规则：

输入	111	110	101	100	011	010	001	000
输出	0	1	1	1	0	1	1	0

为了分辨最简单的细胞自动机，我们需要给每个细胞自动机起一个名字，这个名字可以都用数字进行表示，这就是“方格宇宙”的编码。我们知道，在一类最简细胞自动机中，它们的所有情况都相同，而仅有规则集不同，所以我们只要根据规则集找出编码的方案就相当于给细胞自动机找到了方案了。

同样的道理，在确定了状态数、邻居半径、维数的细胞自动机中，指定了规则集的编号就相当于找到了这个自动机。因此，在后面我们可以通过编号来识别自动机。而这编号的方案有两种，一种编出来的码比较长，但是能给所有的自动机命名，一种比较简捷但不能给所有的自动机命名。下面我会说自动机 24 号，或者 56 号，你要清楚这仅仅是它们的名字，而这两个细胞自动机的规则不一样。一般在没有特殊说明情况下都是用长编码。这些都是细节问题，但在此说明有助于我们后面的讨论。下一部分要讨论如何给细胞自动机编码，属于技术性的问题，不感兴趣的读者完全可以跳到（4）。

（3）“方格宇宙”的编码*

显然，对于最简单的一类细胞自动机，任何一组规则的输入都是上表所列的 8 种情况，而不同的仅仅是输出部分。我们把第一个规则集的输出部分写成：00110101，第二组规则集的输出部分写成：01110110，显然这两个输出不同，但它们都是一个长度为 8 的二进制字符串。我们可以断言任何一个长度为 8 的二进制串都应该对应一组规则。反过来讲，所有可能的规则集都能写为长度为 8 的二进制串，这一共有 2^8=256 种可能，也就是说规则集一共仅有 256 个。我们知道，任意一个二进制串都对应一个十进制的数字，比如 00110101 对应的十进制数是 53。所以，采用这种用 10 进制数字编码的二进制串方法，我们就把规则集进行编码。这种编码的方法很有效，因为任何细胞自动机的规则都逃不出这套编码方案，但是下面我们将看到这种方法有些麻烦，所以我们还常常使用另外一套编码的方法。

如果细胞自动机的邻居半径不是 1，那么上述编码方案的长度将会急剧增加。例如一个邻居半径为 2 的细胞自动机的所有可能输入就是 2^5=32 种，因而根据上面的原理，我们的编码也需要 32 位长，这显然不实际了，因此我们应该能简化该编码方案。考虑这样的规则集，其中每一条规则的输入都仅仅考虑邻居区域的状态数值之和。比如在最简细胞自动机中，某细胞的邻居区域的三个细胞状态是 011，则 0+1+1=2，这样输入就是 2。如果是 010 呢，那么 0+1+0=1，该输入就是 1。这样所有的输入可能仅仅有 4 个包括 0,1,2,3。我们把所有的输入列出，按照类似上面的方法就给每个输入指定输出的状态就得到了该编码。比如一条规则的输入输出对应如下：

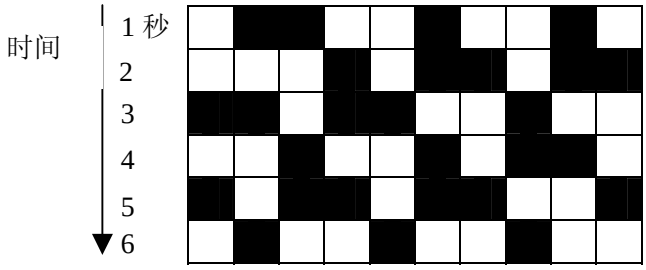
输入	0	1	2	3

输出	1	0	0	1

把 1001 也就是 9 作为该细胞自动集的编码。当邻居半径增大到 2 的时候，所有可能的输入是 5 个，所以编码长度是 5，显然比刚才简化多了。因而这种编码方案为短编码方案。

（4）“方格宇宙”的动态行为

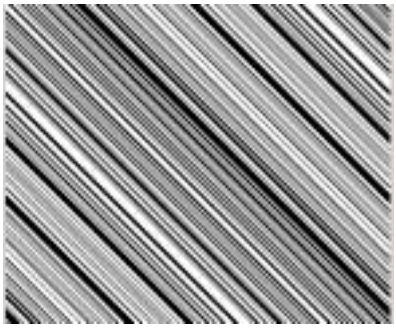
对于一维的情况，我们假设所有的方格都分布在一条直线上，并且直线的长度为 300，也就是说有 300 个方格在这条直线上。我们用黑色的格表示直线上 1 状态的方格，用白色的格表示 0 状态的方格。那么一条断续的横线就是当前所有细胞状态的一种分布。这些方格随着时间变化，就形成了不同的横线。我们把这些随着时间变化的线纵向拼在一起形成了一个网格区域。其中纵轴表示时间的流逝（往下为正），横轴为“方格宇宙”在对应时刻的状态，就能得到一幅图像：



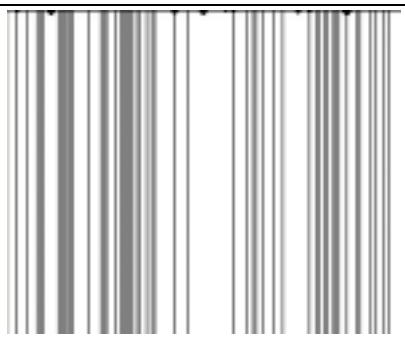
这个方格的每一行都是某一个时刻细胞自动机的状态。因而从上到下数第 1、2、3、4、5、6 行可以分别表示第 1、2、3、4、5、6 秒的细胞自动机状态。因此这里的一个平面的图案就是细胞自动机在时间上的发展动态。

下面我们用程序 onedimension 探索一些最简单的一维细胞自动机的动态图案(注意这里的细胞自动机都采用左右相邻的周期型边界处理)。在状态数为 2，邻居半径 1 的最简情况下分别挑选几种典型的动态情况示于下图（下方的数字是细胞自动机的编码）：

224 号

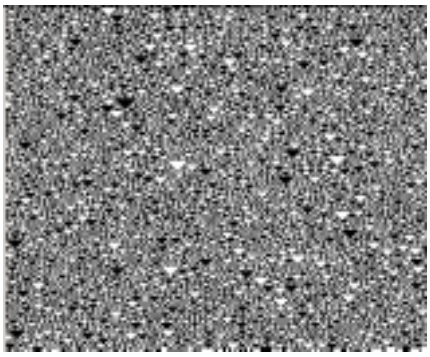


208 号

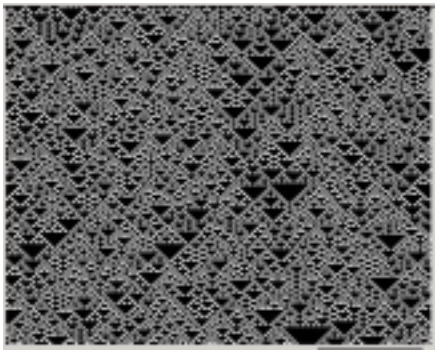


132 号

203 号



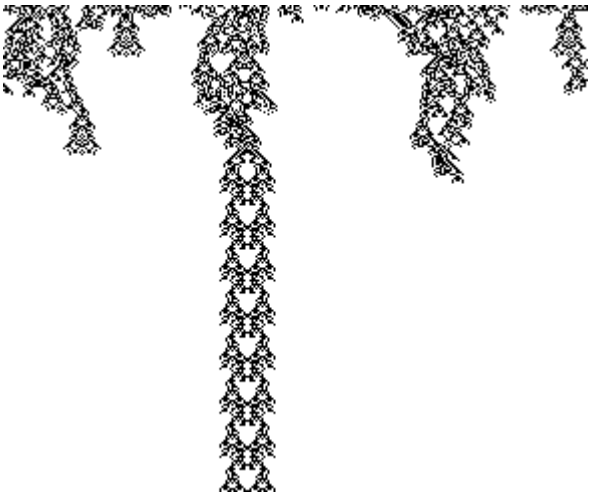
150 号



151 号

我们将这些细胞自动机分成 3 类。上图中的 224 号、132 号和 203 号是一类；208 号细胞自动机是一类，150 号和 151 号是一类。观察 224 号细胞自动机，从上而下出现了一些细胞，之后就逐渐变成了全白色，也就是说经过几个时间步的运行后，细胞自动机全部变为了固定状态 0（也就是白色的方格），并再也不变化了。而 132 号和 203 号细胞自动机都是变成了几个竖线。不要忘了每一行就是某一时刻细胞自动机的一个状态，因此在竖向上能够形成一条竖线就说明这个细胞的状态在时间轴上没有变化。所以 132 号、203 号与 224 号都是一类，它们被吸引到了一个固定的状态。再看 208 号细胞自动机，它是若干条斜的线。由于我们的边界是循环的，因此可以预言，经过若干个时间周期的运行以后，细胞自动机又回复到了原来的状态，因而这样的细胞自动机是循环的。两个相同状态之间经历的时间步长为这种细胞自动机的周期。再看 150 号和 151 号细胞自动机，他们显然既没有固定的周期也没有被吸引到一个点，它们是出于一种混乱的、无序的状态，我们称这种状态为**混沌**状态。通过反复的运行最简细胞自动机程序我们不难发现，所有的 256 种细胞自动机都能被归为这三类：固定值、周期循环、混沌之一。

我们可以猜想，是不是所有的细胞自动机的动态行为就这三种类型呢？让我们把探索的疆域扩大到稍微复杂一点的情况，我们考虑状态数为 2，邻居半径为 2（也就是说每个细胞都有 4 个邻居，左右两边各两个），仍然是一维的情况。在这样的细胞自动机中除了上面叙述的三种类别依然存在外，我们还发现了另一种类型，请看它们的运行图：



20 号（按照短编码方案）

52 号（按照短编码方案）

这两个细胞自动机的动态运行图竟然如此怪异，就好像一棵倒挂的葡萄藤。这种葡萄藤是一种复杂的结构，它既不等同于完全的随机，又没有固定的循环的迹象。这种复杂结构正是我们感兴趣的一种类型，因为它既没有被吸引到固定的点或周期状态而变得死板，又没有因为随机而过于活跃；它既保证了一定的流动活性，同时又能产生具有“记忆性”的结构。该运行情况显然不同于前面叙述的三种类别，所以我们称其为**复杂型**。继续运行各种参数的一维细胞自动机，我们发现几乎所有的一维细胞自动机运行的动态行为都能被划分为这四类情况。

综合上面的讨论，我们把我们的“方格宇宙”归为四种类别，它们分别是：

I、 固定值型：“方格宇宙”经过若干步运算便停留在一个固定的状态；

II、 周期型：“方格宇宙”在几种状态之间周期循环；

III、 混沌型：“方格宇宙”处于一种完全无序随机的状态，几乎找不到任何规律；

IV、 复杂型：“方格宇宙”在运行的过程中可能产生复杂的结构，这种结构既不是完全的随机混乱，又没有固定的周期和状态。

上面我们仅仅就一维细胞自动机的情况作了介绍，二维细胞自动机也无非就是这 4 种情况之一。其实我们想一下，前面介绍的“生命游戏”属于哪种类型呢？当然应该是第 IV 种。只有复杂的类型才会给我们带来永恒的新奇。

下面我们自然会思考一个问题，为什么“方格宇宙”会有这几种类型？它们之间有没有什么联系呢？

6、混沌的边缘

上一节我们给“方格宇宙”分成了四大类，下面就要讨论这四个类别之间的关系。为了更好的看清楚细胞自动机的动态行为，我们选用 4 个状态{0,1,2,3}，邻居半径为 2（一共 4 个邻居）的一维细胞自动机来讨论，因为这种细胞自动机包含了所有的四种类别。

我们知道，在给定了状态集{0,1,2,3}，邻居半径 2 的一维情况下，细胞自动机的规则集决定了它们的不同。每一个细胞自动机的规则集都可以看成是一张大的转换表，形如：

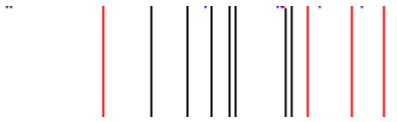
输入	01203	03120	12231	13301	12332	……
输出	1	0	2	3	0	……

其中每个输入的 5 位数字串中，中间的一个表示当前细胞的 t 时刻的状态，两边的数字都是它的邻居状态，而输出则对应当前细胞在 t+1 时刻的状态。表中一共有 4^5=1024 项，这其中有些输出项为 0 状态，有些不为 0，我们把所有输出项为 0 的个数记为 n_q。那么我们可以定义参数：$\lambda=(4^5-n_q)/4^5$，这个参数反映了一组规则中转换成非 0 状态的比例。显然，根

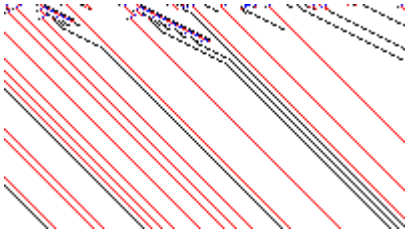
据给定的λ我们可以得到很多的规则表，因此我们可以随机的在这些规则表中选择一个。比如令λ=0.5，那么我们可以随机的生成一个规则组转换表，表的输出部分0状态占据了一半的位置，其他的位置由1,2,3这几个数随机的填充。

下面看看根据参数λ的取值不同，细胞自动机的动态行为如何变化。

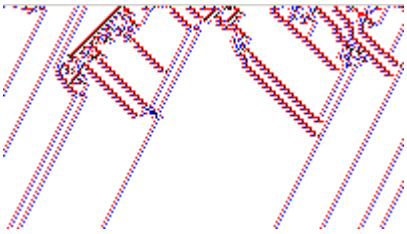
λ=0



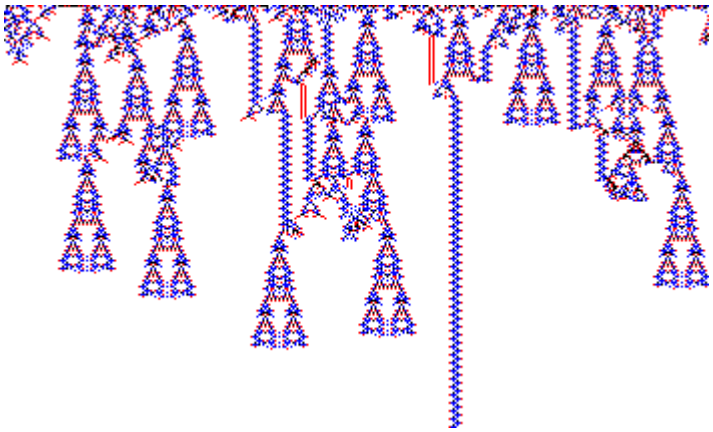
λ=0.1



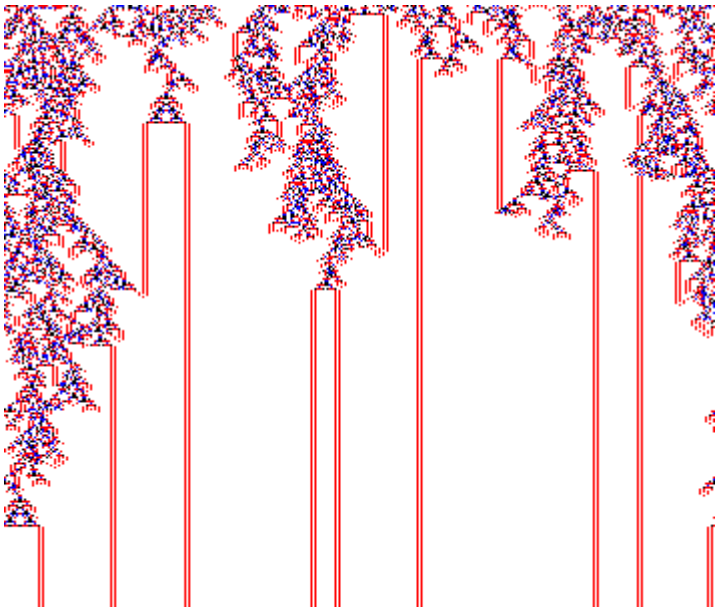
λ=0.2



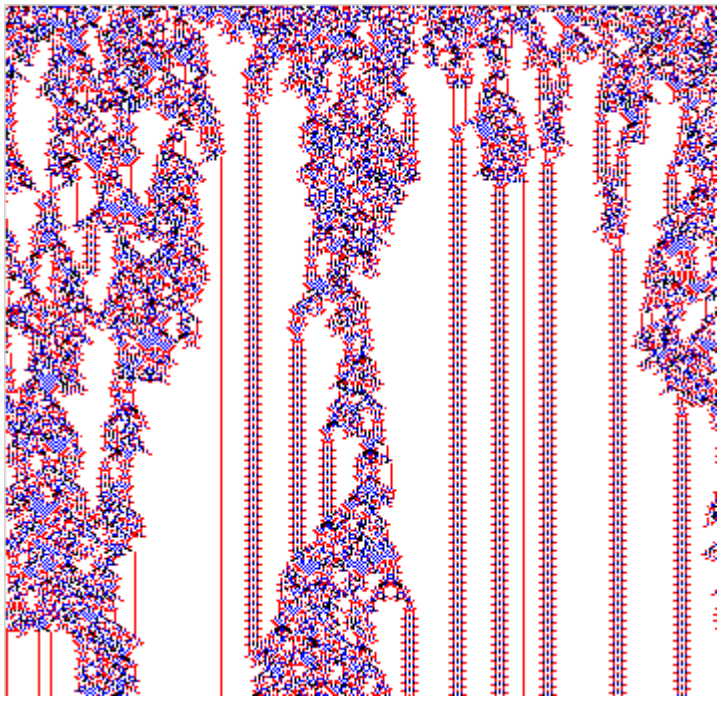
λ=0.3



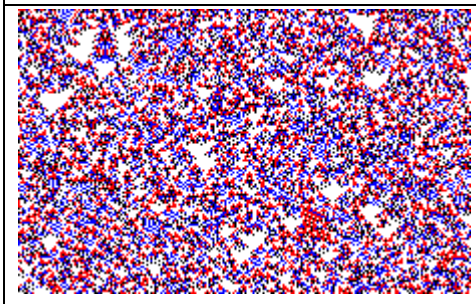
λ=0.4



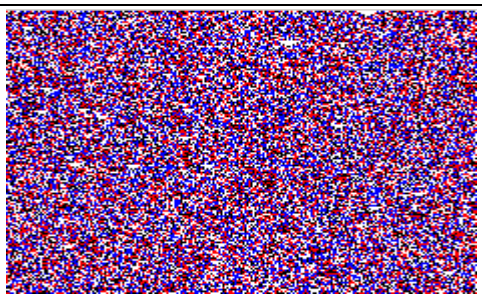
λ=0.45



λ=0.5



λ=0.6



λ=0.7

当λ=0~0.1，所有的细胞被吸引到一种固定的状态，这相当于我们上一节叙述的第一类“方格宇宙”；

λ=0.2~0.3，系统在一些固定的状态之间周期的循环，这相当于第二类“方格宇宙”，但是λ=0.3 的细胞自动机比λ=0.2 的在开始的时候具有更复杂的结构；

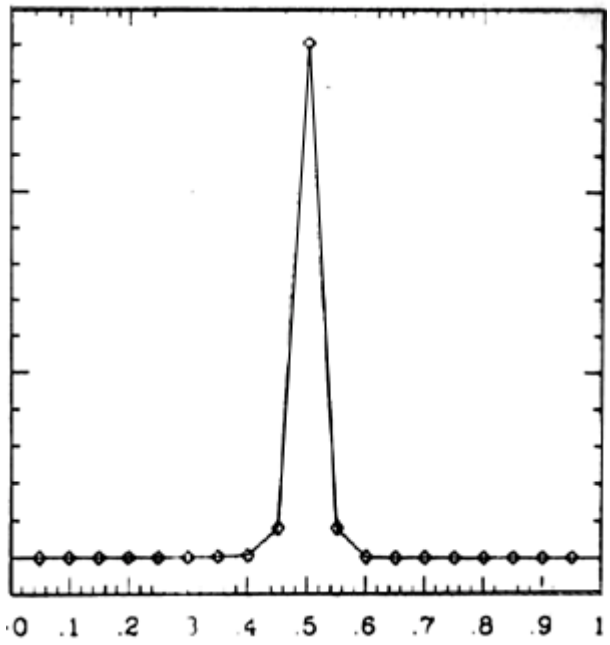
λ介于大约 0.3 到 0.6 之间的时候，会出现相当复杂的结构，如图所示。这些结构既不属于固定的周期或者固定值，也不属于完全的随机，因此这些细胞自动机属于第四类“方格宇宙”即“复杂型”。并且，随着λ的增长，复杂结构的维持时间也会变得越来越大；

λ>=0.6 的时候，复杂的结构消失，系统将被吸引于一种完全随机的混沌状态。

由于在实验中，规则是根据λ随机产生的，因此我们在这里说明的动态行为随λ的变化性质仅仅是一种大致的分类。

根据这些试验，我们不难得出，随着λ的增大，细胞自动机展现出来的结构将逐渐变得复杂，当λ介于一个中间值的时候动态行为会达到最大的复杂性，然后随着λ的进一步增大复杂结构就逐渐被随机结构所取代。

如果把复杂性和λ的数值画在一个坐标系下，我们能得到下面的图：



横轴是λ数值，纵轴是复杂性

根据λ的连续变化能够得到四种细胞自动机之间的过渡转化图景：

I->II->IV->III，即：固定点->周期->复杂->混沌

因此我们说，复杂的结构诞生于混沌的边缘。混沌的边缘是什么东西？它是一种处于凝固的周期状态与活跃的混沌之间的一种过渡过程，或者我们称其为“相变过程”。所谓的“相变”就是指系统从量变到质变的飞跃。就像煮开水，当温度达到100度左右的时候，水会突然沸腾，这种状态就是相变，因为从此水由液态变成了气态。

“方格宇宙”系统的连续变化过程就好像水的固、液已经固态到液态之间的的变化过程。I和II两种状态可以被看作是固态，就像冰一样凝固在一起非常有秩序但同时也没有活性。细胞自动机的第III类型就象是液态的水：完全的流动、随机，没有一个时刻能停留下来，然而由于这类系统过于松散，它也不可能产生有价值的结构。第IV类细胞自动机就刚好存在于从固态的冰到液态的水转变的瞬息之间这么一个狭小的空间里。在这里，复杂的结构形成了神奇的王国，你会不断地看到若干水分子结合成有趣的结构与秩序，但同时这些结构和秩序永远不会被冻结，它们会偶尔被破坏，但新的结构马上又会生成。这样的状态被“人工生命”之父郎顿称为混沌与秩序的边缘。科学家们已经对固体、液体的性质研究的比较清楚了，然而对于固体到液体转变这样一种相变的过程则仍然没有认识足够清楚，原因就在于这样的状态具有太多复杂的结构，我们很难预言它的具体性质。第IV类细胞自动机也是这样，下一时刻我们的“方格宇宙”会是怎样的情况？我们除了按照该“方格宇宙”的“物理规律”运行这个宇宙外别无它法，因为复杂的细胞自动机的行为不能预言。

我们可以把混沌边缘的概念推广，也就是把秩序、周期这些动态的情况看作是一种凝固的吸引力，它保证了系统能够固定于某一种结构；而另一方面，随机、混沌则形成了另一种张力，它使得系统趋于不稳定，但同时为系统提供了创新的动力。那么仅仅当这两种力处于一种恰到好处的平衡态的时候，也就是系统处于混沌的边缘条件下，该系统才会更加有活力，并且演变得越来越复杂。

生命从何处来？智能如何产生？人们为什么会创造有组织的同时又具有创新性的社会结构？答案是这些复杂的系统、复杂的结构来自于混沌的边缘。只有当生命所处的环境既不太“热”，即没有太大的动荡产生完全随机的混沌状态，同时又不能太“冷”，以至于所有活动都过于死板，这样才能孕育真正的生命。因此地球上的生命正是诞生于混沌的边缘这条狭

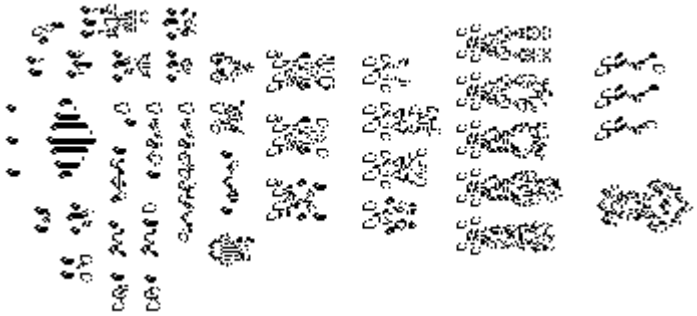
窄的夹缝中。再考虑一个人的发展。如果一个人每天都在做同样的事情，从不尝试新鲜的生活，那么这个人所处的环境就过于死板最后将陷于一种机械循环的状态（第 I、II 类细胞自动机）。反过来，如果这个人过于涣散，总在尝试不同的新鲜事情，从来不会停下来静静的思考和沉淀，那么这个人就会过于灵活而也会一事无成。一个人只有处在“混沌边缘”的状态才能既产生学习、进化的动力，又会静下心来让所学的东西凝固成有价值的知识，从而创造辉煌的成果。再考虑一个国家，闭关锁国肯定不能发展，系统将会变成一潭死水，反过来过于开放则根本不会形成这个国家这个民族的凝聚力也就失去了它们的个性，所以社会也要不停地把自己推向混沌边缘的状态才能不断的发展下去。

系统为什么总要处于“混沌边缘”的状态呢？比如生命吧，静止于某种固定的状态不是挺好的么？为了适应多变的环境，生命必须不断的进化而变得复杂，而要想变得复杂就必须让自己处于混沌边缘的状态。按照达尔文进化论的观点，不适应环境的生命体就会被大自然淘汰了，并不是每个生命体都有意识的要让自己越来越复杂，而是因为过于简单的生命不能适应环境了，于是它们被淘汰掉了，所以剩下来的仅有那些相对复杂的生命体。科学家曾用一批细胞自动机当作生命体在一个虚拟的自然环境中不断的进化，结果发现，仅有那些不断的趋于混沌边缘的细胞自动机留下来，而其他的细胞自动机逐渐被淘汰出局了。

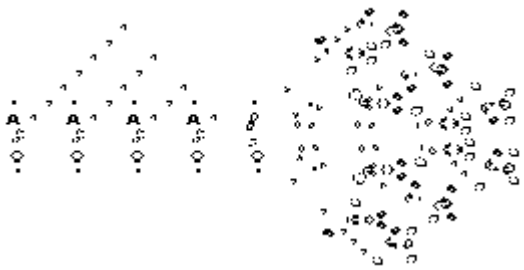
对于细胞自动机的分类以及混沌边缘的概念不仅仅适用于一维细胞自动机，对于二维甚至多维的细胞自动机仍然适用。显然我们熟悉的“生命游戏”也正是处于一种“混沌边缘”的状态。经计算，“生命游戏”对应的 λ=0.25。我们下面再来讨论“生命游戏”，从而更加深刻的理解处于“混沌边缘”的“方格宇宙”。

7、“生命游戏”与通用计算机

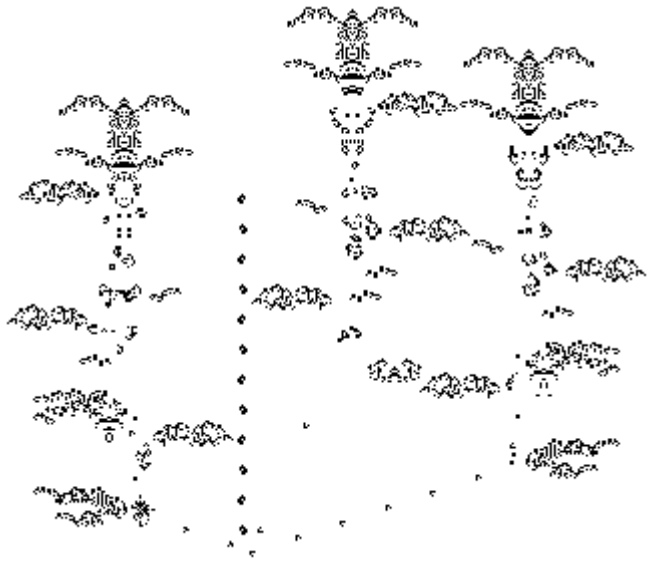
“生命游戏”的规则集、状态集、边界条件等等都是固定的，仅有初始状态可以变化，我们现在就来探索一下，不同的初始状态能够得到什么样的动态行为呢？请到 gameoflife 处玩这个游戏，这里为你准备了各式各样的初始条件。



看，这是由若干个细胞构成的一条往左游动的鱼，它的构形是由若干黑色的方块组成，并且按照生命游戏的两条简单规则不停的演化着造成鱼不停地往屏幕左方移动。你可以通过加载 lifep 文件夹下面的 agua50f.lif 文件得到该结构。

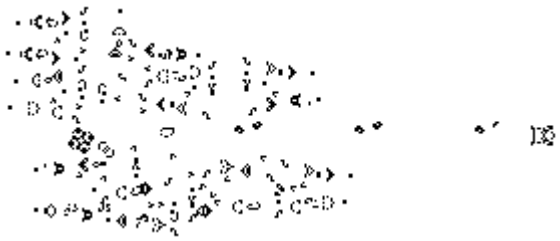


这是一个播种机，右边的好像鱼头一样的东西快速的往右边移动着，并且在他的尾部留下了一长串“滑翔机”，这些“滑翔机”相互作用形成了新的结构就像是种子里诞生出来了婴儿。在 lifep 文件夹下的 breeder2.lif 可以看到。

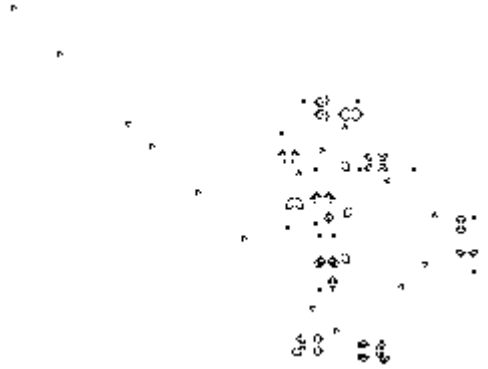


这像三驾喷气式飞机拖着长长的尾巴在空中呼啸而过，并且不断的往上移动着。加载 Lifep/forerake.lif 可以看到。

这是一个火箭腾空而起，尾巴还拖着长长的火焰并逐渐消散。这是文件 Lifep/linepuf.lif 的情景。



这是一座工厂，“滑翔机”工人们在工厂中的各个部门之间不停的往来穿梭着，并且工厂还有很多的大机器，有些机器在制作着“滑翔机”工人，有些则在不停的消耗着原材料。在右边这个工厂还会往外发射物质，就好像是它正往外输出产品。Lifep/Sawtooth.lif



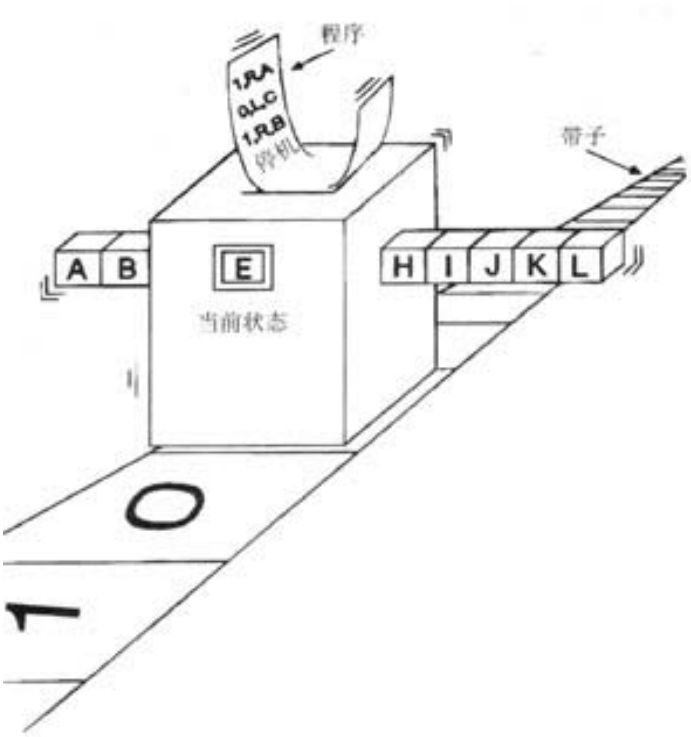
这个机构不算复杂然而却有惊人的功能：产生一个随机的“滑翔机”序列。多个“滑翔机”在这个机构的左下角不断产生出来，但是产生两个“滑翔机”的时间间隔却没有固定的值，因此这个序列呈现出混沌的性质。

在 Life32 这个程序中，你还能看到更多各式各样的“生命游戏”构形，足够让你眼花缭乱了，不要忘记所有这些都是那两条简单规则运行的结果。另外，在 Life32 中，你还能设计自己的简单结构，看看它将会有怎样的演化结果。你还可以把两个或多个有趣的结构放到一起，比如让一群鱼在一起游泳，把两个“工厂”放到一起组合成更大的工厂等等，所有的一切仅仅受陷于你的想象力！

也许在此之前你还在怀疑“生命游戏”这个简单的游戏除了能给我们带来一些好看的图案以外还能给我们带来什么呢？看到上面的那些产生于“生命游戏”的结构之后也许你的好奇心被激发起来了，原来这个简单的游戏中还有那么多的有趣东西呢。进一步，你大概会问：这个东西在功能上的极限是什么？这个“方格宇宙”能否与我们真实的宇宙相媲美呢？嗯，估计你的怀疑的声音要更大些，“方格宇宙”一定没有这个真实世界复杂，因为毕竟在这个真实世界中还有一台可怜的计算机，在这个计算机上正在跑一个叫做“生命游戏”的可怜程序，这一切能出现在“方格宇宙”里么？那如果在这个生命游戏所在的“方格宇宙”中诞生了一台计算机会怎样呢？同样的，这台虚拟的计算机也可能正在跑一个在它那个世界里面的虚拟的“生命游戏”！

是不是太“科幻”了？但这未必不能实现。实际上，人们已经在原理上证明了“生命游戏”完全有可能产生一台计算机的虚拟结构，而且我们计算机能完成的计算过程都可以在“生命游戏”中完成！也就是说，该断言为在“生命游戏”中再虚拟牵套一个“生命游戏”提供了理论上的可能。下面让我们一起大致领略一下证明的过程。

为了理解这种虚拟计算机的可能性，我们首先要理解究竟什么是计算机？这个问题大概很可笑，然而你使用的计算机由于过于复杂反而不易于我们的理解。我们需要的并不是一台具体的计算机，而是需要知道计算机的计算过程的本质是什么？幸好，早在 20 世纪 30 年代的天才科学家阿伦.图灵已经给我们提供了答案。他从理论上证明了所有的可计算的过程都可以在一台被后人称为“图灵机”的最简计算机上完成。下图给出了一台“图灵机”的样子，注意我们关心的仅仅是这台机器的信息处理功能，而不是它的硬件。



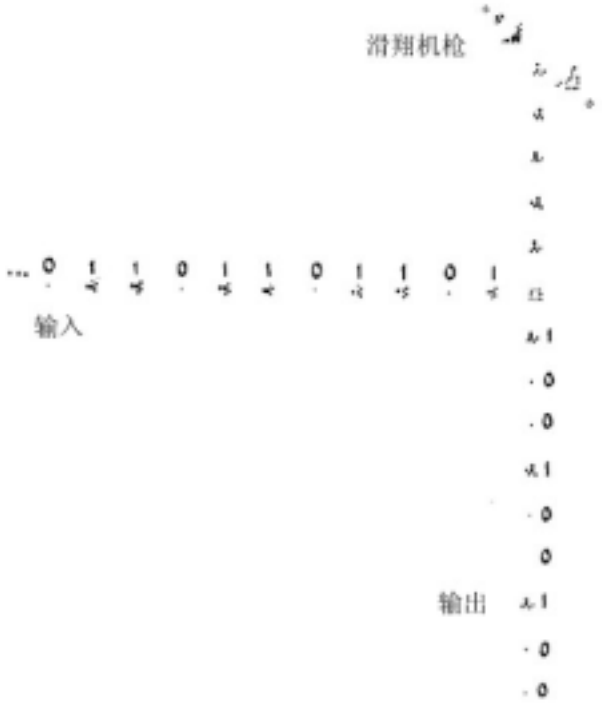
一个图灵机的典型部分包括：一条无限长的纸带，上面记载了数据，一些程序（指令的集合），一个存储器。程序按照当前的存储器上的状态和从纸带上读入的输入信息进行操作。我们不必关心它的技术细节，只需要知道虽然我们现在使用的计算机再复杂也逃不出图灵机的能力范围。所以图灵机又叫做“通用计算机”，也就是说宇宙中所有的计算机都逃不过图灵机的能力范围。

人们已经证明“生命游戏”是和图灵机等价的，但是证明过于专业。在此我们换一种方法让你理解究竟“生命游戏”如何等价于“通用计算机”。在此之前，我们先要明白任意一个复杂的计算过程都是由若干对数据操作的简单指令运算组合而成的。这些数据可以被描述成二进制位串，更简单的计算指令又逃不过简单的对二进制数的“与”、“或”、“非”等运算。“与”运算相当于一台机器，如果给它输入两个二进制串 01001,10101，则这个机器就输出 00001 也就是把两个二进制串对应位置上的两个数取最小的以形成新串。“或”运算则是把这两个数每个位置上的数字取大。而“非”运算则是让一个二进制串每一位上的 0 或者 1 翻转，也就是原来某一位置上是 0 的话，则经过“非”运算就变成 1，这样把 01001 输入“非”运算这台机器，那么输出就是 10110。这几个简单的运算就是构造复杂计算过程的基本砖块，下面我们只需要说明在“生命游戏”中能够产生这些基本砖块就可以了。我们先来看下面这个“生命游戏”中的结构：

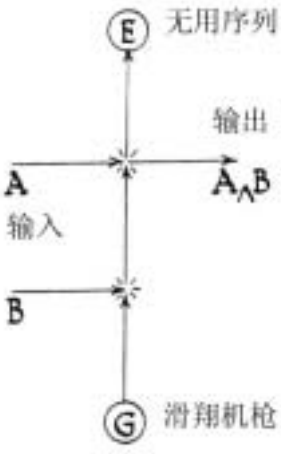
这个结构能不停的产生“滑翔机”，就好像一个机关枪。在 life32 的游戏中加载 Lifep/makegun.lif 可以看到。我们可以把任意一个滑翔机看作是一个信号，因此它在“方格世界”中可以传播出去。这样，这个“滑翔机枪”就可以产生一列信号，如果在屏幕的右

下方有一个信号接收器，那么它将接到一列信号。我们把这个序列中的每个滑翔机看作是信号“1”，这样右下角接收到的序列就应该是“111111……”，假如某个“滑翔机”在传输的过程中发生意外没有爬到接收器，那么接收器接收到的信号序列就会在某个位置缺少一个信号，我们把这种缺少的情况记为 0，这样接收到的序列就是“101111……”。由此可见，“滑翔机”序列就是虚拟宇宙中的虚拟信号。另外，“滑翔机枪”还可以被用作虚拟计算机的时钟。

下面再考虑这样一个结构：



这个结构能够完成前面叙述的“非”运算功能，因而它等价于一个“非”运算器。输入的滑翔机序列通过与滑翔机枪输出的序列相互作用后会在下端的输出一个与原来的输入序列正相反的“非”运算序列。通过观察“生命游戏”我们知道，两个“滑翔机”在互相垂直的方向上相遇会抵消掉，因此原来的 1 就会变成 0。而如果输入端是一个 0 信号也就是没有“滑翔机”在某个时刻发射过来，那么“滑翔机枪”发射的“滑翔机”就不会被抵消，而传送到输出端的就是一个 1 信号。同样的道理，下面的结构可以产生“与”运算的效果：



输入两个序列 A 和 B，首先 B 先与滑翔机产生的序列通过对消产生 B 的非序列，然后

非序列再与 A 序列对消从而在右端得到 A 和 B 的“与”序列（记为 A∧B）。假如 B 序列是 10011，那么经过 G 的相互作用后就会得到序列 01100，这个序列与 A 再相互作用。假设 A 是 11100，也就是在前三个时间步内有“滑翔机”，后两个时间步内没有“滑翔机”，这样在 A 与非 B 序列相遇的点上，第一个时间步非 B 没有“滑翔机”经过，因而，会在输出端得到一个“滑翔机”即得到 1。第二个时间步有“滑翔机”与 A 序列的“滑翔机”对消，因而输出端得到的是 0，以此类推，……，最后输出端得到的序列就是 10000，这刚好是 A∧B 的结果。同样的道理“并” 运算也可以制作出来，这里不再叙述了。

这样，简单的“与”、“并”、“非”运算都可以在“生命游戏”中得到，那么这些运算的合成也可以通过把上面介绍的结构合成起来组成一个庞大的网络来完成。另外还有存储功能也可以用生命游戏中的结构来完成，这里不再说明。也就是说通用计算机的任何一种复杂的运算都能通过这些简单的运算组合排列形成，所以我们不难相信我们手头上的计算机能够做的事情从原则上都能让“生命游戏”来做，也就是说“生命游戏”与通用计算机在功能上是等价的。

我们运行的“生命游戏”这个软件过程当然也可以在“生命游戏”提供的虚拟宇宙平台上运行，这表明“生命游戏”构成了硬件基础，而“生命游戏”这个软件在这个“生命游戏”的特殊硬件上得到了重生。也就是说“生命游戏”之中又诞生了一个“生命游戏”，而同样的逻辑一样适用于这个虚拟的“生命游戏”，就是说在“生命游戏”的“生命游戏”中又会诞生一个“生命游戏”……。这意味着什么？世界套世界么？想想当你把两面镜子相对而放的时候会看到什么？一个无穷仅的镜子的序列，而且是一个镜子套一个镜子！同样的事情可以发生在“生命游戏”中，“自己包含自己”的无穷结构诞生了，这也许是我们所处的宇宙以及所有可能的虚拟宇宙中的一条深刻的原理，然而现在的科学理论还没有解释这其中的深层含义，也许它就在那里等待着我们去发现呢！