SIAM REVIEW
Vol. 41, No. 3, pp. 594–604

Integer Programming and Conway's Game of Life[*]

Robert A. Bosch[†]

Abstract. This article presents integer programming formulations for finding interesting patterns in Conway's game of Life, with accompanying exercises and solutions.

Key words. integer programming, game of Life, cellular automata

AMS subject classifications. 90C10, 90C27, 90C90, 90D99, 68Q80, 00A08

PII. S0036144598338252

1. Introduction.

In the late 1960s, John Horton Conway invented a one-player game called Life. To begin the game, the player places checkers on some of the squares of an infinite checkerboard. The player then follows a small set of simple rules to create a new pattern of checkers from the current pattern. By applying the rules again and again, the player produces a sequence of patterns. Often, these sequences are both aesthetically pleasing and mathematically interesting. In 1970 and 1971, Martin Gardner popularized Life by making it the subject of two columns in *Scientific American* [3, 4]; see also [5]. Since then, a great number of individuals—some of them professional mathematicians, and some of them amateurs—have contributed to an understanding of the game [1, 2, 6, 7].

In this module, we explore the role that integer programming can play in the search for interesting Life patterns. (See [8] for an introduction to integer programming.) In sections 2 and 3, we describe the rules of Conway's game, introduce some terminology, give some examples, and present two integer programming formulations of the *maximum density still-life problem*. Section 4 consists of exercises in modeling with binary variables. Section 5 contains solutions.

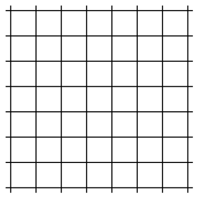
This module has been used by the author in a course that provides junior and senior mathematics majors with an introduction to linear optimization. The material from sections 2 and 3 is presented in a single-period lecture. At the end of the lecture, one or more problems from section 4 are assigned. Student response has been very favorable. In fact, several students have gone on to pursue long-term independent projects on integer programming and Life.

2. The Rules of the Game.

The game of Life is a single-player board game. The playing pieces are checkers, all the same color. The board is a checkerboard that extends to infinity in each direction. We refer to the squares of the board as *cells*, and we consider a cell to have eight *neighbors*: the eight cells that share one or two corners with it. See Figure 1.

[*]Received by the editors May 4, 1998; accepted for publication (in revised form) January 14, 1999; published electronically July 27, 1999.
http://www.siam.org/journals/sirev/41-3/33825.html

[†]Department of Mathematics, Oberlin College, Oberlin, OH 44074 (bobb@cs.oberlin.edu).



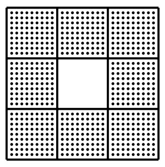


Fig. 1

2.1. The Initialization Phase. To begin the game, the player constructs an initial pattern, placing checkers in some of the cells of the board. Which cells receive a checker and which ones don't is entirely up to the player. We refer to a cell with a checker in it as *alive* or *living* and a cell without one as *dead*.

The initialization phase is the only part of the game in which the player makes any decisions.

2.2. The Main Phase. The player executes this phase over and over again. Each time, the player follows the three simple rules listed below and produces a new pattern from the current one.

- If a cell currently has exactly two living neighbors, then its state—living or dead—must be the same in the new pattern as it is in the current pattern.
- If a cell currently has exactly three living neighbors, then it has to be alive in the new pattern.
- If a cell currently has fewer than two, or more than three, living neighbors, then it has to be dead in the new pattern.

We refer to these rules as the *status-quo rule*, the *birth rule*, and the *death rule*. Note that the death rule is really two rules: a *death-by-isolation rule* and a *death-by-overcrowding rule*.

We refer to the initial pattern as the $t = 0$ pattern, the pattern that results from the first execution of the main phase as the $t = 1$ pattern, and so on. We say that a cell is alive (dead) at $t = i$ if it is alive (dead) in the $t = i$ pattern.

2.3. Examples. In Figures 2, 3, and 4 (and in all subsequent figures that display Life patterns), cells that contain solid circles are healthy (currently alive, and staying alive), while cells that contain ringed circles are unhealthy (currently alive, but about to die). Empty cells are of course dead. Cells that contain tiny dots are dead but are about to become alive.

The Life object displayed in Figure 2 is a *still life*, a pattern that never changes. At every time t, every living cell has two or three living neighbors, and every dead cell has fewer than three or precisely six living neighbors. So every living cell stays alive, and every dead cell remains dead.

The object displayed in Figure 3 is a period-2 *oscillator*. The even-time patterns (the patterns at $t = 0$, $t = 2$, $t = 4$, and so on) are all identical, as are the odd-time patterns. The object displayed in Figure 4 is a period-4 *spaceship* that "moves" one row up and one column to the right every four time units.

3. Finding Dense Still Lifes. Recall that a still life is a Life pattern that never changes. Each cell is either immortal or eternally dead. Suppose that we wish to

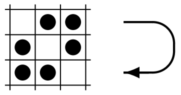


Fig. 2

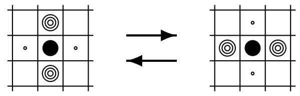


Fig. 3

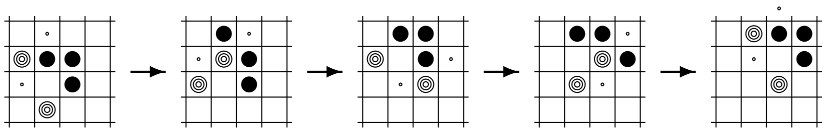


Fig. 4

find a maximum density still life that can be constructed inside a specified region. In this section, we present two integer programs for solving this problem. We define the density of a still life over a region to be the number of immortal cells in the still life divided by the total number of cells in the region.

3.1. Variables. In order to construct a still life inside a region R, we must decide, for each cell e in R, whether e will be immortal or eternally dead. We therefore let, for each cell e in R,

$$x_e = \begin{cases} 1 & \text{if } e \text{ is immortal,} \\ 0 & \text{if } e \text{ is eternally dead.} \end{cases}$$

3.2. Constraints. For each cell in R, we need to impose constraints that enforce the death-by-isolation, death-by-overcrowding, and birth rules.

For still lifs, the death-by-isolation rule states that cells with fewer than two immortal neighbors must be eternally dead, and the death-by-overcrowding rule states that cells with more than three immortal neighbors must be eternally dead. It is very easy to construct constraints that enforce these rules. If we define $N(e)$ to be the set of cells in R that are neighbors of e, then the inequalities

$$\text{DI}(e) \qquad 2x_e - \sum_{f \in N(e)} x_f \le 0$$

and

$$\text{DO}(e) \qquad 3x_e + \sum_{f \in N(e)} x_f \le 6$$

guarantee that the death-by-isolation and death-by-overcrowding rules hold at cell e. Incidentally, the reason why the coefficient of x_e is 3 and the right-hand side is

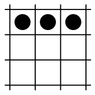


Fig. 5

6 in the death-by-overcrowding constraint (instead of 5 and 8, respectively) is that if a cell e had more than six living neighbors at time t, then at least one of those neighbors must have had more than three living neighbors at time t and therefore, by the death-by-overcrowding rule, would be dead at time $t+1$. In other words, in a still life, eternally dead cells can have no more than six immortal neighbors.

For still lifes, the birth rule states that cells with exactly three immortal neighbors must be immortal too. The birth rule is harder to model than the death-by-isolation and death-by-overcrowding rules. One approach involves using three constraints and two auxiliary 0-1 variables to enforce the birth rule at a cell e:

$$\mathrm{B}_1(e) \qquad \sum_{f \in N(e)} x_f \le 2 + 4a_e,$$

$$\mathrm{B}_2(e) \qquad \sum_{f \in N(e)} x_f \ge 4 - 4b_e,$$

and

$$\mathrm{B}_3(e) \qquad a_e + b_e \le 1 + x_e.$$

Note that $\mathrm{B}_1(e)$ forces the auxiliary variable a_e to equal 1 when e has three or more immortal neighbors, $\mathrm{B}_2(e)$ forces the auxiliary variable b_e to equal 1 when e has three or fewer immortal neighbors, and $\mathrm{B}_3(e)$ forces x_e to equal 1 when both a_e and b_e equal 1.

A second approach uses 56 constraints per cell but no auxiliary variables. Each of these constraints prohibits exactly one of the $\binom{8}{3} = 56$ different ways in which the birth rule can be violated. Note that in order for the birth rule to be violated at cell e, e must be eternally dead and have exactly three immortal neighbors, as in Figure 5.

To prohibit these violations of the birth rule, we impose constraints of the form

$$\mathrm{B}'(e,S) \qquad -x_e + \sum_{f \in S} x_f - \sum_{f \in N(e)-S} x_f \le 2,$$

where S is a 3-element subset of $N(e)$. Note that $\mathrm{B}'(e,S)$ demands that if all three of the cells in S (in Figure 5, the cells in the top row) are immortal, then so is e or at least one of the cells in $N(e) - S$.

The only other constraints we need are for ensuring that the still life we construct in R stays inside R. If e is a cell outside of R and has exactly three neighbors, f, g, and h, that belong to R, then we must impose the "boundary" constraint

$$x_f + x_g + x_h \le 2.$$

3.3. The Full Formulations. Our discussion of the variables and constraints makes it clear that to find a maximum density still life that can be constructed in a

Table 1

		Formulation IP$_1$			Formulation IP$_2$		
n	Gap	Nodes	Iterations	Seconds	Nodes	Iterations	Seconds
4	1.88	215	847	0.4	38	280	0.3
5	0.50	125	969	0.4	1	107	0.2
6	3.61	168,212	1,344,178	734.6	1,758	19,037	48.2
7	1.67	1,950	35,545	18.4	172	3,662	13.5
8	2.44	ran out of memory			3,996	99,653	500.3
9	4.58	ran out of memory			232,450	4,000,071	33,1630.6

specified region R, we can solve one of the following two integer programs:

$$\begin{array}{lll} \text{IP}_1 \quad \text{maximize} & \displaystyle\sum_{e\in R} x_e & \\ \quad\quad \text{subject to} & \text{DI}(e),\ \text{DO}(e) & \forall e \in R, \\ & \text{B}_1(e),\ \text{B}_2(e),\ \text{B}_3(e) & \forall e \in R, \\ & \text{boundary constraints for } R, & \\ & x_e, a_e, b_e \in \{0,1\} & \forall e \in R, \end{array}$$

or

$$\begin{array}{lll} \text{IP}_2 \quad \text{maximize} & \displaystyle\sum_{e\in R} x_e & \\ \quad\quad \text{subject to} & \text{DI}(e),\ \text{DO}(e) & \forall e \in R, \\ & \text{B}'(e,S) & \forall e \in R : \forall S \subseteq N(e) : |S| = 3, \\ & \text{boundary constraints for } R, & \\ & x_e \in \{0,1\} & \forall e \in R. \end{array}$$

Note that IP$_1$ has considerably fewer constraints than IP$_2$ but three times the number of variables.

3.4. Comparing the Formulations. We compared IP$_1$ and IP$_2$ on $n \times n$ regions. To solve the integer programs, we used version 4.0.9 of CPLEX's Mixed Integer Optimizer. Each time, we set CPLEX's *lowercutoff* parameter equal to $\lceil n^2/2 \rceil$. (By doing so, we were instructing CPLEX to ignore all still lifes with density less than 1/2.) We kept all other parameters at their default values.

Our results (summarized in Table 1) suggest that IP$_2$ is a more easily solved formulation than IP$_1$. The 9×9 square was the largest square region R for which IP$_2$ was tractable; the 7×7 square was the largest for IP$_1$. Moreover, IP$_2$ required fewer branch-and-bound nodes, fewer iterations of the simplex method, and less CPU time on all regions tested. The superiority of IP$_2$ might be surprising to readers who have had little or no exposure to integer programming. After all, the two formulations have the same "relaxation gap" (the difference between the optimal value of the linear programming relaxation and the optimal value of the integer program), and IP$_1$ has considerably fewer constraints than IP$_2$. But IP$_2$ has two-thirds fewer variables than IP$_1$. The fact that IP$_2$ has considerably fewer variables than IP$_1$ is perhaps the main reason for its better numerical performance.

But even with IP$_2$, we were unable to solve anything larger than the 9×9 problem. We did find some valid inequalities that, when added to IP$_2$, enabled us to solve the 10×10 problem. But even with these valid inequalities, we were unable to

$$\underset{\mathrm{DI}(e)}{\begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline -1 & 2 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array}} \le 0 \qquad \underset{\mathrm{DO}(e)}{\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 3 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}} \le 6 \qquad \underset{\mathrm{B}'(e,S)}{\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline -1 & -1 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array}} \le 2$$

Fig. 6

solve anything larger than the 10×10 problem. Some possible avenues for future research include searching for strong valid inequalities for IP_2, constructing specialized algorithms for solving IP_2, and developing and analyzing heuristics.

3.5. Comments and Figures. We conclude this section by presenting some figures. Figure 6 demonstrates that the constraints of IP_2 can be displayed in a simple tabular form. (Imagine that we lay one of these tables of numbers on top of R in such a way that the nine cells of the table overlap nine cells of R. By doing this, we are selecting nine cells of R and, consequently, nine variables of IP_2. The selected variables are the ones that appear in the constraint. Their coefficients are given by the numbers in the table.) Note that the rightmost constraint in Figure 6 prohibits the configuration displayed in Figure 5.

Figure 7 displays some maximum density still lifes that can be constructed in square regions. As mentioned earlier, we found the 10×10 pattern by appending some simple valid inequalities to IP_2. Figure 8 displays some high-density still lifes that have 90-degree rotational symmetry. They were obtained by solving a simple modification of IP_2.

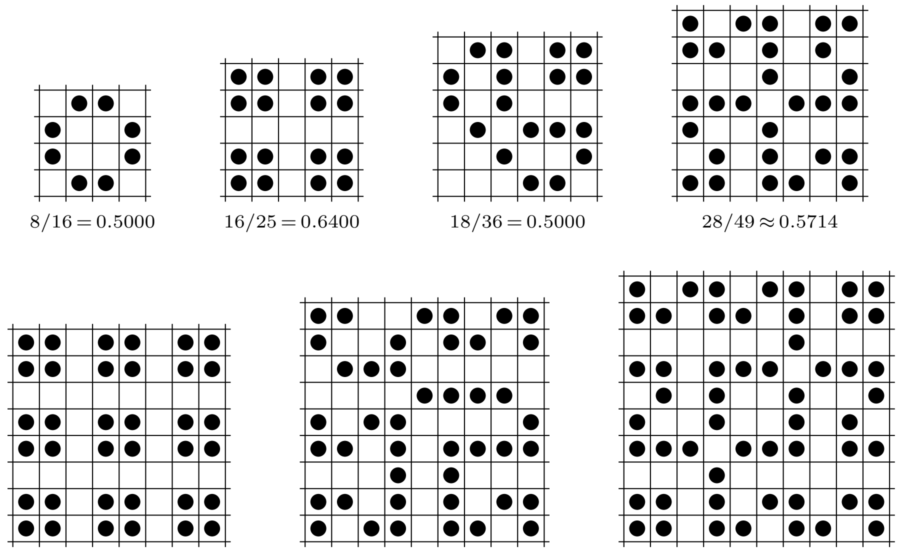


Fig. 7

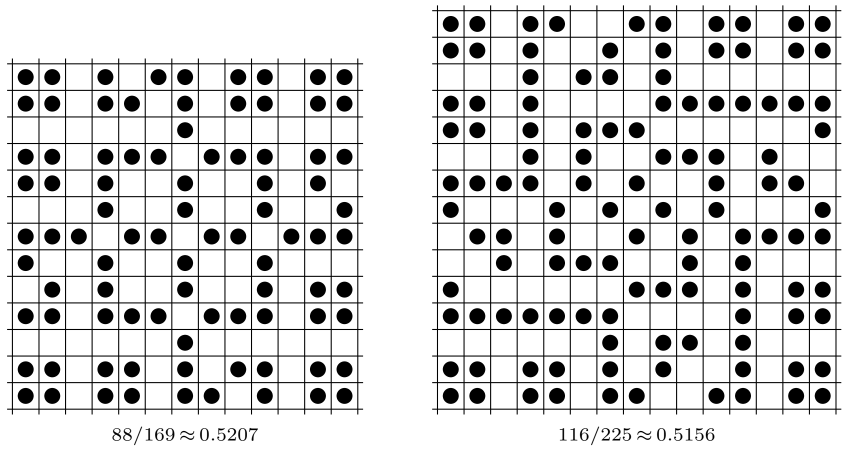


Fig. 8

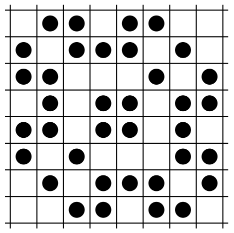


Fig. 9

4. Exercises. *Exercise* 1. There are many variants of Life. In 3-4 Life, a cell is alive in the new pattern if and only if it has three or four living neighbors in the current pattern. Modify IP_1 so that it could be used to find maximum density still lifes in 3-4 Life. (Figure 9 displays a maximum density 3-4 still life that can be constructed in an 8×8 region.)

Exercise 2. Modify IP_2 so that it could be used to find maximum density still lifes in 3-4 Life.

Exercise 3. Modify IP_1 so that it could be used to find maximum average density period-2 oscillators in Life. (Figure 10 displays a period-2 oscillator that can be constructed in an 8×8 region.)

Exercise 4. Modify IP_2 so that it could be used to find maximum average density period-2 oscillators in Life.

Exercise 5. Devise an integer program that could be used to find period-2 spaceships that move to the left. Generalize this formulation so that it applies to period-m spaceships that move to the left.

Exercise 6. Suppose that you encounter a person playing Life. This person shows you his or her $t = 1$ pattern and asks you to determine the $t = 0$ pattern. See Figure 11. Of course, this is impossible. After all, there exist *Garden-of-Eden* patterns that have no immediate predecessors. And any pattern that isn't a Garden-of-Eden pattern has an infinite number of immediate predecessors. So suppose that you wish

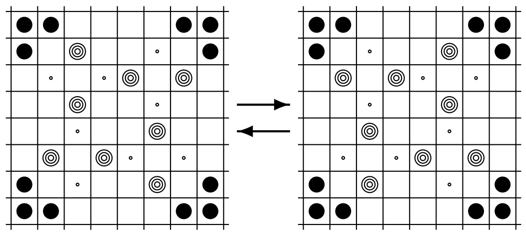


Fig. 10

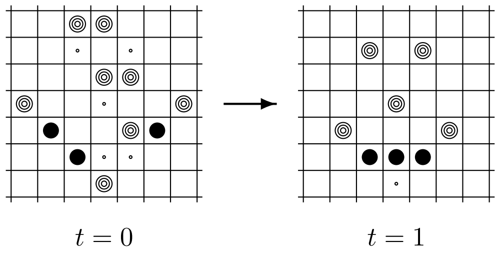


Fig. 11

to find, amongst all patterns that *could* have been the $t = 0$ pattern, a pattern with the fewest number of living cells. Formulate this problem as an integer program.

5. Solutions.

Solution to Exercise 1. To enforce the rules of 3-4 Life at cell e, include the constraints

$$3x_e - \sum_{f \in N(e)} x_f \le 0,$$

$$4x_e + \sum_{f \in N(e)} x_f \le 8,$$

$$\sum_{f \in N(e)} x_f \le 2 + 6a_e,$$

$$\sum_{f \in N(e)} x_f \ge 5 - 5b_e,$$

and

$$a_e + b_e \le 1 + x_e.$$

The first two constraints enforce the death rules. The third forces a_e to equal 1 when e has three or more immortal neighbors, and the fourth forces b_e to equal 1 when e has four or fewer immortal neighbors. The fifth constraint forces x_e to equal 1 when both a_e and b_e equal 1.

Solution to Exercise 2. To enforce the birth rule of 3-4 Life without using auxiliary variables, include the constraints

$$-2x_e + \sum_{f \in S} x_f - 2 \sum_{f \in N(e)-S} x_f \leq 2 \qquad \forall S \subseteq N(e) : |S| = 4.$$

Solution to Exercise 3. For each cell e in R and for each time $t \in \{0, 1\}$, let

$$x_{e,t} = \begin{cases} 1 & \text{if } e \text{ is alive at time } t, \\ 0 & \text{if } e \text{ is dead at time } t. \end{cases}$$

To enforce the death rules at cell e during the transition from the $t = 0$ pattern to the $t = 1$ pattern, include the constraints

$$2x_{e,1} - \sum_{f \in N(e)} x_{f,0} \ \leq\ 0$$

and

$$5x_{e,1} + \sum_{f \in N(e)} x_{f,0} \ \leq\ 8.$$

(Similar constraints are needed for the transition from the $t = 1$ pattern to the $t = 0$ pattern.) To enforce the birth rule at cell e during the transition from the $t = 0$ pattern to the $t = 1$ pattern, introduce auxiliary 0-1 variables $a_{e,0}$ and $b_{e,0}$ and include the constraints

$$\sum_{f \in N(e)} x_{f,0} \ \leq\ 2 + 6a_{e,0},$$

$$\sum_{f \in N(e)} x_{f,0} \ \geq\ 4 - 4b_{e,0},$$

and

$$a_{e,0} + b_{e,0} \ \leq\ 1 + x_{e,1}.$$

Finally, to enforce the status-quo rule at cell e during the $t = 0$–to–$t = 1$ transition, introduce auxiliary 0-1 variables $c_{e,0}$ and $d_{e,0}$ and include the constraints

$$\sum_{f \in N(e)} x_{f,0} \ \leq\ 1 + 7c_{e,0},$$

$$\sum_{f \in N(e)} x_{f,0} \ \geq\ 3 - 3d_{e,0},$$

$$c_{e,0} + d_{e,0} + x_{e,0} \ \leq\ 2 + x_{e,1},$$

and

$$c_{e,0} + d_{e,0} + x_{e,1} \ \leq\ 2 + x_{e,0}.$$

Note that $c_{e,0}$ equals 1 when e has two or more living neighbors at $t = 0$, that $d_{e,0}$ equals 1 when e has two or fewer living neighbors at $t = 0$, and that $x_{e,0} = x_{e,1}$ when both $c_{e,0}$ and $d_{e,0}$ equal 1.

Solution to Exercise 4. Here, to enforce the birth rule at cell e for the $t = 0$–to–$t = 1$ transition, include the constraints

$$-x_{e,1} + \sum_{f \in S} x_{f,0} - \sum_{f \in N(e)-S} x_{f,0} \leq 2 \qquad \forall S \subseteq N(e) : |S| = 3.$$

To enforce the status-quo rule at cell e during this transition, include the constraints

$$3x_{e,1} - x_{e,0} - \sum_{f\in N(e)} x_{f,0} \le 0$$

and

$$-x_{e,1} + x_{e,0} + \sum_{f\in S} x_{f,0} - \sum_{f\in N(e)-S} x_{f,0} \le 2 \qquad \forall S \subseteq N(e) : |S| = 2.$$

Note that the first of these status-quo constraints prohibits all configurations in which cell e is dead at $t = 0$, has two (or fewer) living neighbors at $t = 0$, and is alive at $t = 1$. The remaining constraints prohibit all configurations in which cell e is alive at $t = 0$, has exactly two living neighbors at $t = 0$, and is dead at $t = 1$.

Note that the modifications of IP_1 and IP_2 presented above do not rule out still lifes. In fact, in every case we tested, the optimal solution was a still life, not a period-2 oscillator. One solution to this problem is to pick a cell e_1 and force it to "blink" by fixing $x_{e_1,0} = 1$ and $x_{e_1,1} = 0$. By forcing different cells to blink, we can obtain different oscillators.

Solution to Exercise 5. Suppose we want to find a period-2 spaceship that moves to the left. Let $r(e)$ denote the cell to the right of cell e. Consider the IP_2-based formulation for finding period-2 oscillators. Leave the constraints for the $t = 0$–to–$t = 1$ transition unchanged. In each constraint for the $t = 1$–to–$t = 0$ transition, replace the variable $x_{e,0}$ with $x_{r(e),0}$.

Solution to Exercise 6. Let A stand for the set of cells that are alive at $t = 1$, and let D stand for the set of cells that are dead at $t = 1$. Let x_e equal 1 if e is alive at $t = 0$ and 0 otherwise. Then we can find a smallest immediate predecessor by solving the following integer program:

$$\begin{aligned}
\min\ & \sum_{e} x_e \\
\text{s.t.}\ & \sum_{f\in N(e)} x_f \le 3 && \forall e \in A,\\
& x_e + \sum_{f\in N(e)} x_f \ge 3 && \forall e \in A,\\
& \sum_{f\in S} x_f - \sum_{f\in N(e)-S} x_f \le 2 && \forall e \in D : \forall S \subseteq N(e) : |S| = 3,\\
& 2x_e + 2\sum_{f\in S} x_f - \sum_{f\in N(e)-S} x_f \le 4 && \forall e \in D : \forall S \subseteq N(e) : |S| = 2,\\
& x_e \in \{0,1\} && \forall e \in A \cup D.
\end{aligned}$$

The first two sets of inequalities, taken together, guarantee that all of the cells that are supposed to be alive at $t = 1$ are in fact alive then. To see why, let e be a cell that is alive at $t = 1$. Note that the inequality

$$\sum_{f\in N(e)} x_f \le 3$$

ensures that e has at most three living neighbors at $t = 0$, while the inequality

$$x_e + \sum_{f\in N(e)} x_f \ge 3$$

ensures that if e is alive at $t = 0$, then e has at least two living neighbors at $t = 0$, and that if e is dead at $t = 0$, then e has at least three living neighbors. By taking the two inequalities together, we come to two conclusions: (1) if e is alive at $t = 0$, then e has either two or three living neighbors at $t = 0$, and (2) if e is dead at $t = 0$, then e has exactly three living neighbors at $t = 0$.

The third and fourth sets of inequalities play a similar role, but for the cells that are supposed to be dead at $t = 1$. To see why, let e be a cell that is dead at $t = 1$. Note that if e_1, e_2, and e_3 are the only living neighbors of e at $t = 0$, then the inequality

$$\sum_{f \in S} x_f - \sum_{f \in N(e)-S} x_f \leq 2,$$

with $S = \{e_1, e_2, e_3\}$, is violated. Also note that if e is alive at $t = 0$ and e_1 and e_2 are the only living neighbors of e at $t = 0$, then the inequality

$$2x_e + 2\sum_{f \in S} x_f - \sum_{f \in N(e)-S} x_f \leq 4,$$

with $S = \{e_1, e_2\}$, is violated.

Acknowledgments. The author is grateful to the referees for their helpful comments.

REFERENCES

[1] E. R. Berlekamp, J. H. Conway, and R. K. Guy, *Winning Ways for Your Mathematical Plays, Volume* 2: *Games in Particular*, Academic Press, London, 1982.

[2] P. Callahan, *Conway's life miscellany web page*, http://www.cs.jhu.edu/∼callahan/lifepage.html, 1998.

[3] M. Gardner, *The fantastic combinations of John Conway's new solitaire game "Life,"* Sci. Am., 223 (1970), pp. 120–123.

[4] M. Gardner, *On cellular automata, self-reproduction, the Garden of Eden and the game "Life"*, Sci. Am., 224 (1971), pp. 112–117.

[5] M. Gardner, *Wheels, Life, and Other Mathematical Amusements*, W. H. Freeman, New York, 1983.

[6] W. Poundstone, *The Recursive Universe: Cosmic Complexity and the Limits of Scientific Knowledge*, William Morrow, New York, 1985.

[7] R. T. Wainwright, *Lifeline: A Quarterly Newsletter for Enthusiasts of John Conway's Game of Life*, Issues 1–11, 1971–1973.

[8] L. A. Wolsey, *Integer Programming*, John Wiley, New York, 1998.