Langton's Ant

Introduction: 2 - state automata

Langton's Ant is an example of a 2 dimensional cellular automaton played on a square grid whose cells can be in one of two states P or Q.

At each move the ant does three things:

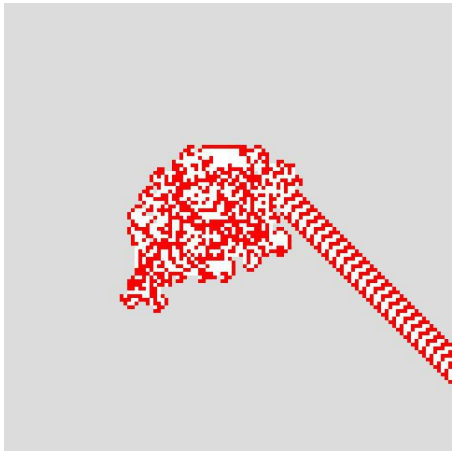
1. Turn left or right according to the state of the cell it is in
2. Toggles the state of the cell
3. Moves forward one square

We can specify this behaviour in the form of a simple table with 2 entries:

P: QL Q: PR

which simply means: if you are on a cell in state P, turn left and change the state to state Q and similarly for the other entry.

The result is shown below



After wandering about for about 10,000 generations, it eventually builds a pathway to heaven.

Now there are 16 possible algorithms but all those beginning PL or PR never change the state of any cell and simple go round in circles. Eliminating all reflections (i.e. those algorithm in which L and R are interchanged leaves just four algorithms of interest, namely

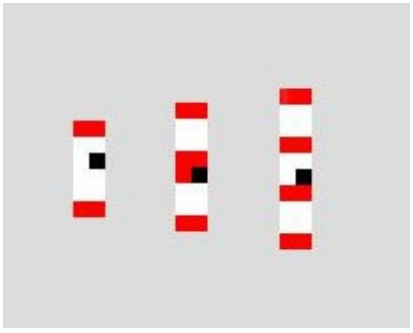
1. P: QL Q: QL
2. P: QL Q: QR
3. P: QL Q: PL
4. P: QL Q: PR

The first and third end by cycling round a 2 by 2 square. The second builds a 4 by 4 square before reaching a 2 by 2 cycle. The fourth is, of course, Langton's Ant.

Some additional interest can be added by allowing the ant to move forward instead of having to turn left or right. Many of these ants simply mow a narrow strip but the following one is interesting:

P: QL Q: PF

and produces the following output:



which shows the state after 28, 63 and 135 generations. Amazingly the ant counts in binary!

Another variant is to introduce a fourth turn option – that of 'turn round and go backwards' but this does not produce any really interesting behaviour. To find some more interesting ants it is necessary first to increase the number of possible cell states.

3 - state automata

If we label the three cell states P, Q and R and the turn options L, R, F and B there are 1728 possible ants ($3^3 \times 4^3$). Consider first the cyclic ants in which state P becomes Q, state Q becomes R and state R becomes P. Eliminating left/right reflections, the number of turn options reduces from 64 to 48 essentially different possibilities which we may put in a table thus:

QL RL PL	QF RL PL	QB RL PL
QL RL PR	QF RL PR	QB RL PR
QL RL PF	QF RL PF	QB RL PF
QL RL PB	QF RL PB	QB RL PB
QL RR PL	QF RR PL	QB RR PL
QL RR PR	QF RR PR	QB RR PR
QL RR PF	QF RR PF	QB RR PF
QL RR PB	QF RR PB	QB RR PB
QL RF PL	QF RF PL	QB RF PL
QL RF PR	QF RF PR	QB RF PR
QL RF PF	QF RF PF	QB RF PF
QL RF PB	QF RF PB	QB RF PB
QL RB PL	QF RB PL	QB RB PL
QL RB PR	QF RB PR	QB RB PR
QL RB PF	QF RB PF	QB RB PF
QL RB PB	QF RB PB	QB RB PB

The cells in grey indicate ants which quickly become cyclic, those in yellow mow a vertical strip, often in binary, those in green build a pathway and those in red behave chaotically.

It is clear on reflection that any algorithm which starts *F is always going to go straight off the board.

We need to consider another possibility. Cell state P must become something else (e.g.Q) otherwise all the cells would forever remain in state P. In order to use state R, state Q must become state R. But state R does not have to return to state P, it can either remain in state R or go to state Q.

The following table shows the latter case:

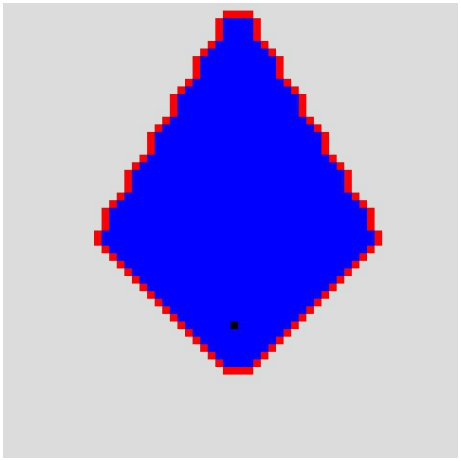
QL RL QL	QF RL QL	QB RL QL
QL RL QR	QF RL QR	QB RL QR
QL RL QF	QF RL QF	QB RL QF
QL RL QB	QF RL QB	QB RL QB
QL RR QL	QF RR QL	QB RR QL
QL RR QR	QF RR QR	QB RR QR
QL RR QF	QF RR QF	QB RR QF
QL RR QB	QF RR QB	QB RR QB
QL RF QL	QF RF QL	QB RF QL
QL RF QR	QF RF QR	QB RF QR
QL RF QF	QF RF QF	QB RF QF
QL RF QB	QF RF QB	QB RF QB
QL RB QL	QF RB QL	QB RB QL
QL RB QR	QF RB QR	QB RB QR
QL RB QF	QF RB QF	QB RB QF
QL RB QB	QF RB QB	QB RB QB

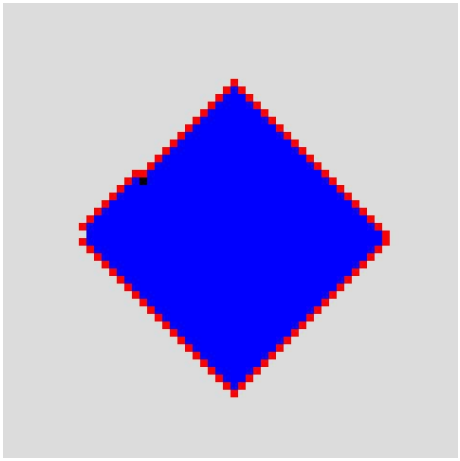
but the former case throws up the odd surprise:

QL RL RL	QF RL RL	QB RL RL
QL RL RR	QF RL RR	QB RL RR
QL RL RF	QF RL RF	QB RL RF
QL RL RB	QF RL RB	QB RL RB
QL RR RL	QF RR RL	QB RR RL
QL RR RR	QF RR RR	QB RR RR
QL RR RF	QF RR RF	QB RR RF
QL RR RB	QF RR RB	QB RR RB
QL RF RL	QF RF RL	QB RF RL
QL RF RR	QF RF RR	QB RF RR
QL RF RF	QF RF RF	QB RF RF
QL RF RB	QF RF RB	QB RF RB
QL RB RL	QF RB RL	QB RB RL
QL RB RR	QF RB RR	QB RB RR
QL RB RF	QF RB RF	QB RB RF
QL RB RB	QF RB RB	QB RB RB

The cases shown in blue generate a solid block of state R cells bordered by a line of state Q cells.

QB RF RL and QB RF RR build a regular diamond shape; QL RR RF (and, of course, QR RL RF) builds a kite shape like this:





but QB RL RR and QB RR RL try to build a square but enter a cycle after 638 generations!

2×2 - state automata

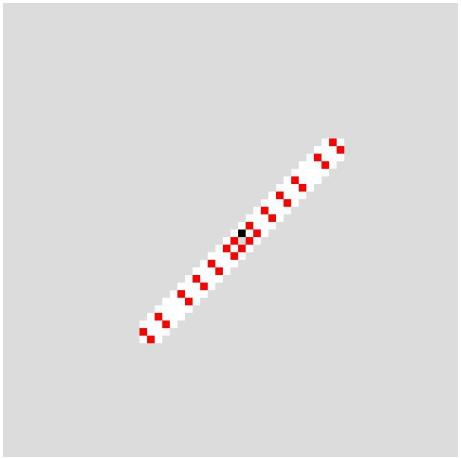
There is a further class of automata called 'turmites' which posses internal states of their own. We shall label these states A, B, C etc. Each entry in the table now consists of three letters. A simple 2×2 turmite which toggles its state and the state of the cell at each generation might have the following definition:

	P	Q
A	BQL	BQR
B	APF	AQL

By permutating L, R, F and B there are 256 of these but most either shoot off to infinity or wander around chaotically. The example above does something a little different. in 46 generations it draws a little castle and then sticks in a loop.

This amazing one counts in binary along a diagonal line:

	P	Q
A	BPR	BQL
B	AQF	APL



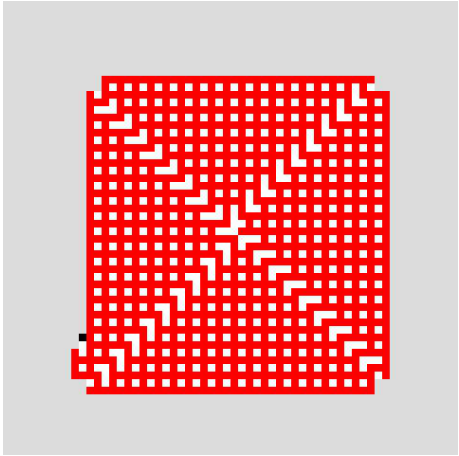
Here is one that draws a square growing ever bigger:

	P	Q
A	AQL	BQR
B	BPL	AQR

And one that draws a particularly pleasing pattern:

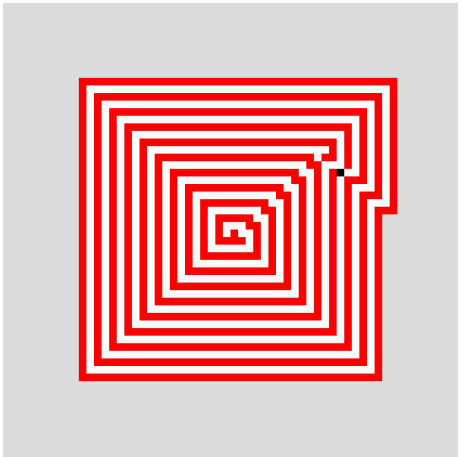
	P	Q
A	BQL	APR
B	APR	BQL

Which looks like this:



Here's one that draws a square spiral:

	P	Q
A	BQR	APL
B	BQL	APR



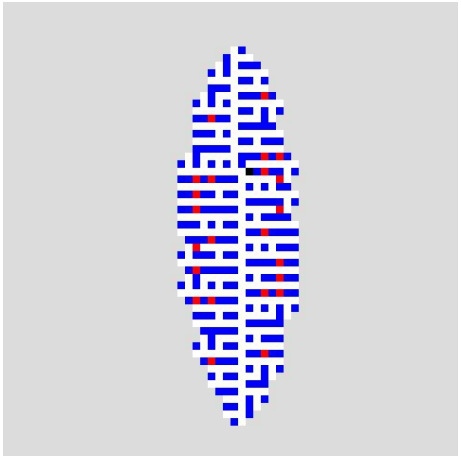
2×3 - state automata

Moving on to automata with 2 internal states and three cell states, the permutations become enormous. The best way to proceed is to try some random combinations. Here are a few which I

have discovered:

The Zulu shield:

	P	Q	R
A	BPR	BQR	AQR
B	BRR	BRR	APL



This one makes a diamond carpet by weaving up and down and round and round..

	P	Q	R
A	BQR	BQR	APR
B	BQL	BRR	AQL

This one makes a square carpet by working round and round in a spiral:

	P	Q	R
A	ARR	BPL	AQL
B	APL	APR	BQL

This one makes a rope with a loop at the end:

	P	Q	R
A	BRR	APB	BPR
B	AQL	BRR	BPB