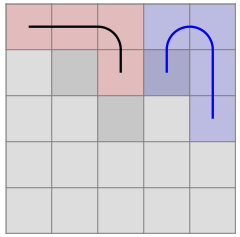
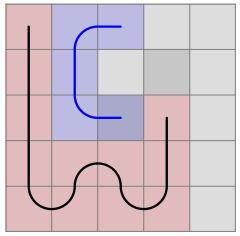
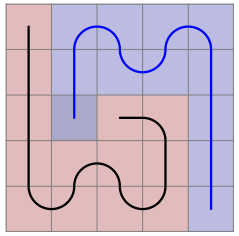
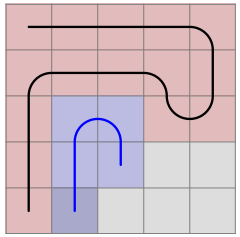


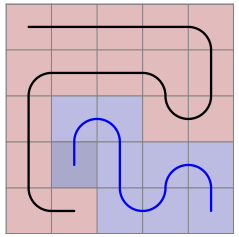
- A 2 x 2 empty area next to the live end that is walled by three sides (2-2-2 long) will have a future line going through along the walls. In this example, the far end is already extended by one step as it had only one option to move.

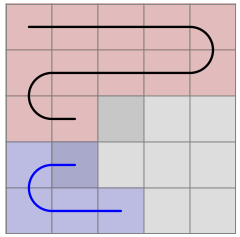




- Future line extension when we step on a future line: The far can be extended if it was 2 distance away from the near end. It can now fill the C-shape.





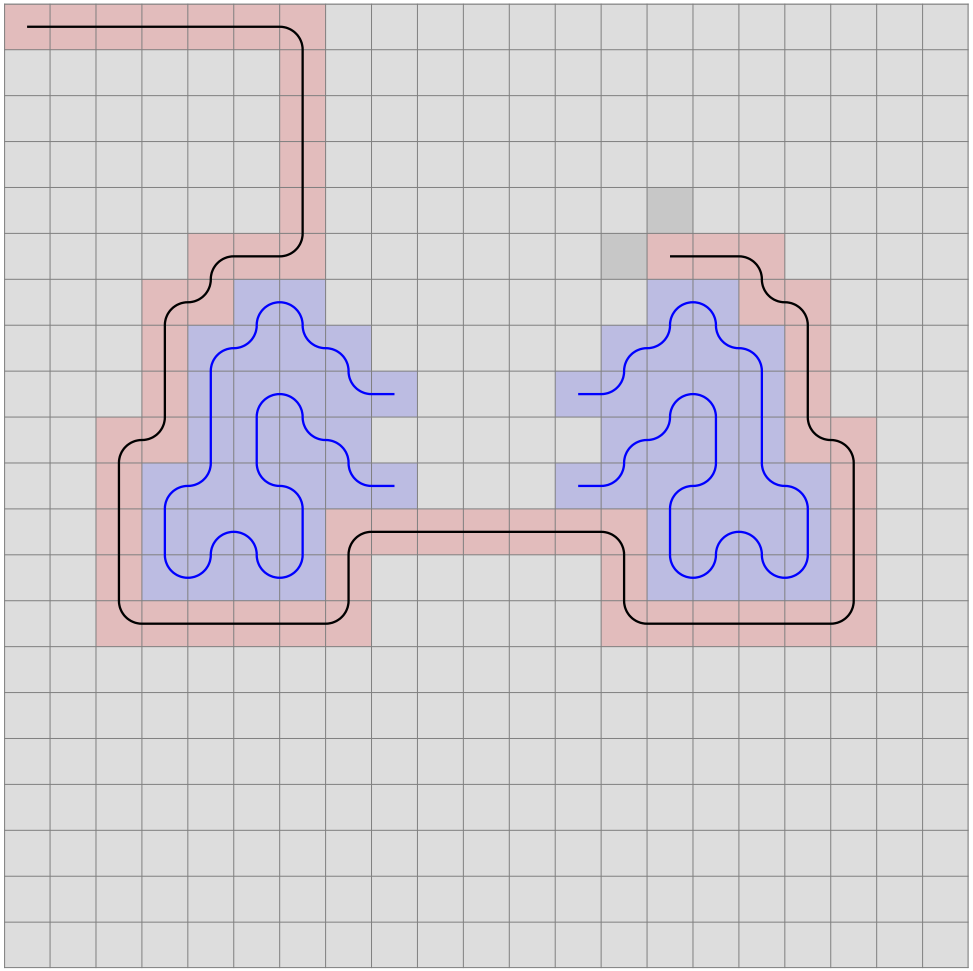


The same goes with 1 x- and y-distance. A C-Shape is not always created in this case.

The one-way labyrinth algorithm

This research aims to solve the following problem:
"Draw a line that goes through an n x n grid (where n is an odd number), passing through each field once. The line has to start from the field at the upper left corner (1,1) and end at (n,n). At any time it is allowed to move left, right, up or down, and it has to randomly choose between the available fields."

At first sight it may look easy. But look at the following example:

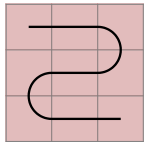


Based on the black line's movement, blue fragments were drawn to indicate a path we have to go through in the future in order to fill the board.
Do you see why the situation is impossible from now on?

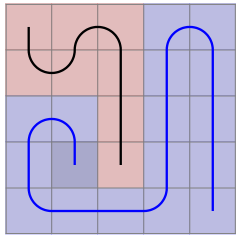
The question is, is there a single rule or a set of rules that will guarantee you can draw a labyrinth of any size? Or do the rules get infinitely complex?

To assist with the research, I have written a computer program. In the beginning, I let it run on a 21 x 21 field, and whenever I noticed a trouble, I coded the solution into it. While you can discover many patterns this way, they will be random and do not help in gaining a fundemental understanding. At one point you will find things get too complex, and you are still far from solving the 21 x 21 board.
That's where a gradual approach comes in.

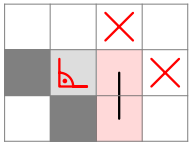
A 3 x 3 area can only be filled in two ways, like this and mirrored:

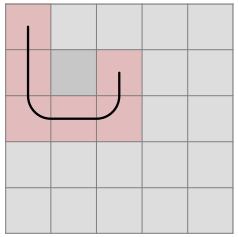


The 5 x 5 requires much more consideration. Whenever it is possible to draw future lines, the program has to be able to do it. The future lines can not only extend at each step but connect too.

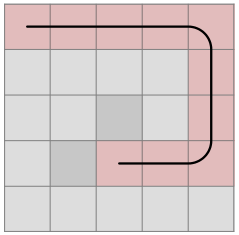


By August 21, 2023 all 5 x 5 scenarios were discovered. The number of walkthroughs are 104.
Here are the things to consider on a grid of this size:

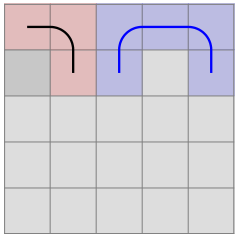




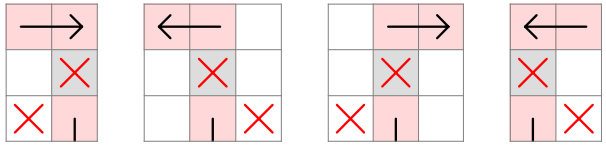
- A single field next to the live end that is walled from two other sides (either by the border or the line) needs to be filled in the next step. I call it C-shape. The pattern is both mirrored and rotated, so that the empty field is straight ahead. To qualify for this rule, the empty field cannot be the end corner. If there is a C-shape, we don't need to check other rules.



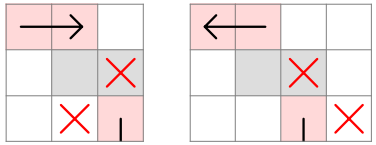
- Movement near the edge: In the example, we cannot step left (3,5), since the (2,5) field is empty.



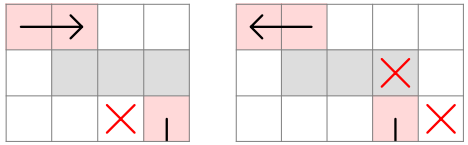
- A 2 x 3 empty area next to the live end that is walled by three sides (2-3-2 long) will have a future line going through along the walls. At the wall next to the main line, its direction is the opposite of the main line, meaning it will go from (3,2) upwards whereas the main line just took a step downwards. How the middle field will be filled is not yet known. Either the near end (the one the main line will go through first) or the far end can fill it.



The gray square means empty field. When the field 2 to straight is taken, its left or right side will be taken too.

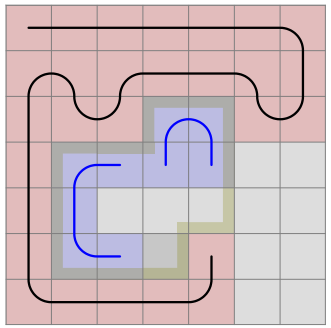


These will only be checked if one of the above 4 situations were not present. (They have to be mirrored, too.)

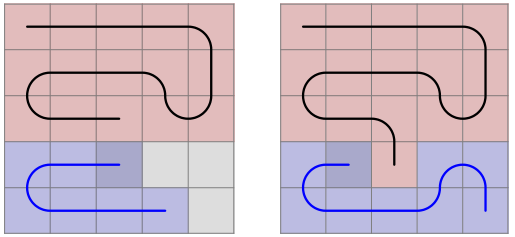


Likewise, these will be not be checked if the previous rules were true.

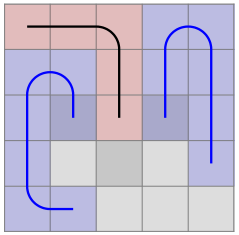
And when none of the 1-distance situations are valid, we check for 2-distance.



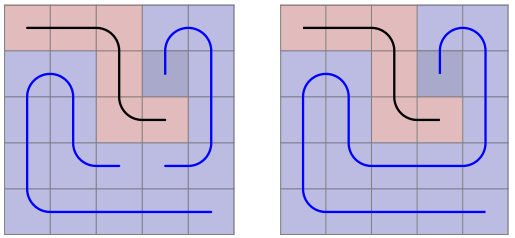
Impair areas can now happen inside the grid, not just on the edge, and the following rules have to be applied:



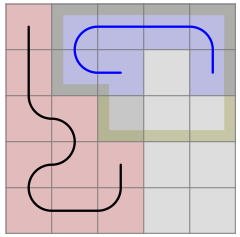
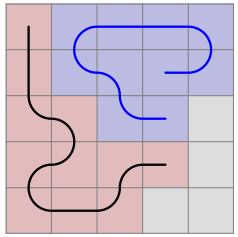
If the far end was near the end corner, it has to choose the other empty field.



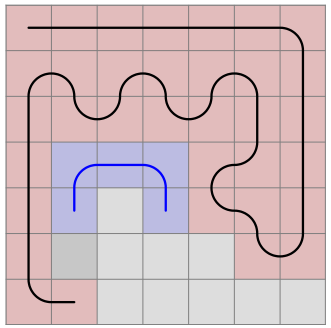
- Future line extension when stepping away: If there was a near end where the main line was in the previous step, it now may have only one choice to move, so it can be extended.



- Future line connection: In this case, the line being stepped on extends until the far end has two options. (When the end corner is one of them, it has to be removed.) Then, the line on the left extends and now has no other option than to connect to the line on the right.

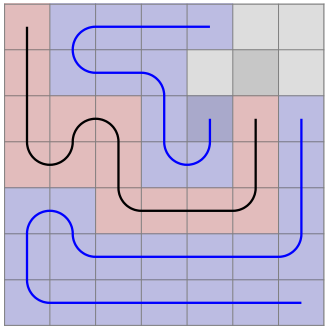
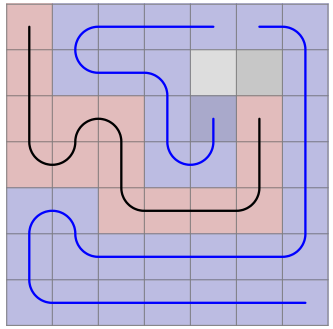



- When we are two distance away from the edge, we need to check if stepping towards it is possible.
It is because if we do so, an enclosed area is created, with one way to go out of it. If that area has an impair amount of cells, it cannot be filled, so we cannot take that step.
The explanation is simple: Imagine if the table was a chess board. In order to step from white to black, you would need to take an impair amount of steps - the color changes at every step. Here, the entry of the area would be (4,3) and the exit (5,3). An impair amount of steps means pair amount of cells.
In the example, you can also say that we cannot step right, because there is a future line start 2 to straight and an end 2 to straight and 2 to right. On 7 x 7, there will be examples where this is the rule we have to apply, because area counting is not getting triggered:

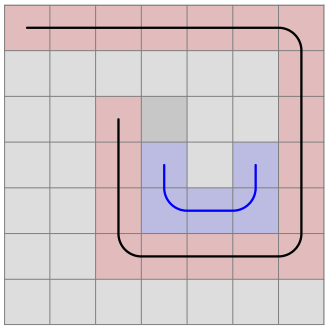


But let's start with the simpler rules:

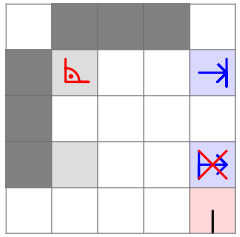
- Future line extension: When a near end is at 2 distance left or right from the live end, it will fill the field between them if the live end steps elsewhere. That's what happened in the 5 x 5 example above before the line failed.

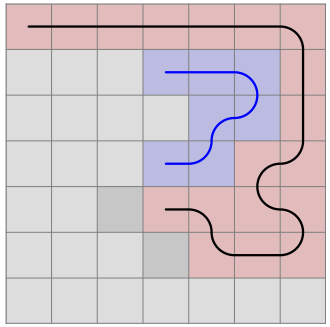



In other situations, there is a 1-thin future line next to the live end that can be extended if its far end is at the corner. Though disabling this rule does not affect the total amount of walkthroughs on a 7 x 7 grid, I chose to include it in the project on the basis that if a future line can be extended, we should do it. It can make a considerable difference. The left picture is without the rule, the right is with it.



- Just like moving near the edge, we need to disable some fields if we are approaching an older section of the main line. In order to determine on which side the enclosed area is created, we need to examine the direction of the line at the connection point.





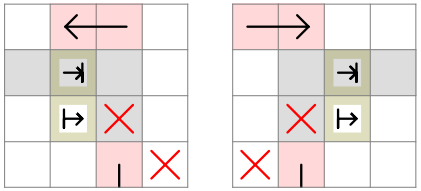
- And these are the remaining size-specific rules. Future 2 x 2 Start End, Future 2 x 3 Start End and Future 3 x 3 Start End.

The program, in fast mode, can run through approximately 100 cases per second, depending on your computer speed. This enables us to discover all 7 x 7 walkthroughs, which is 111 712.
It is equal to what is described in the Online Encyclopedia of Integer Series (Number of simple Hamiltonian paths connecting opposite corners of a 2n+1 x 2n+1 grid).

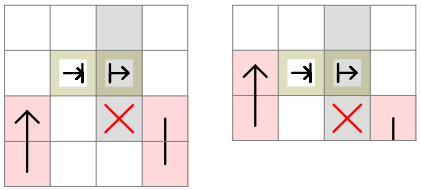
As the sizes grow, it will be impossible to run through all cases with one computer in a reasonable time. In order to discover the patterns, we need to run the program randomly.

Is it possible to develop an algorythm that works for all sizes? The edge-related and area-counting rules are universal, but the size-specific rules get more and more complex. Can you define them with one statement?

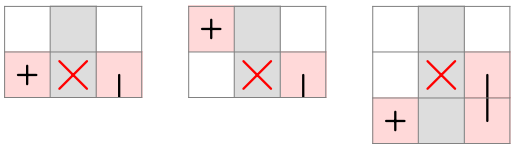
I have made statistics about how many random walkthroughs you can complete on different grids using the 7 x 7-specific and the universal rules before running into an error. Based on 1000 attempts, here are the results:
9: 19.5
11: 5.7
13: 2.6
15: 1.2
17: 0.7
19: 0.4
21: 0.2



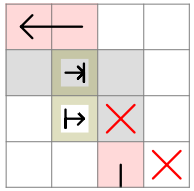
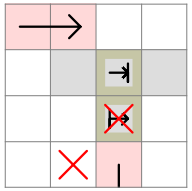
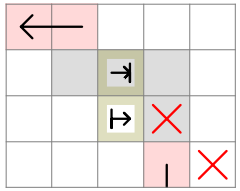
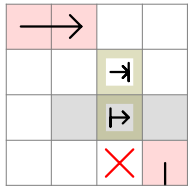
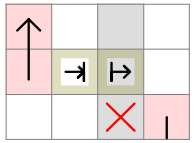
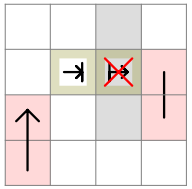
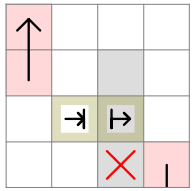
The procedure is similar to the the straight 2-distance rule. The only difference is that we count the area starting and ending at the marked fields. In the first, the direction of the circle is left, in the second right.
Besides mirroring them, we also have to rotate them both counter-clockwise and clockwise.
But we do not need 12 of such rules. Taking the first, the live end cannot come from the left, because the area parity was already checked in the previous step, and now we just added 2 fields to it. It can come from the right, and then there is naturally only one field we might have to disable.
Here are the representations of the two scenarios for the left side:



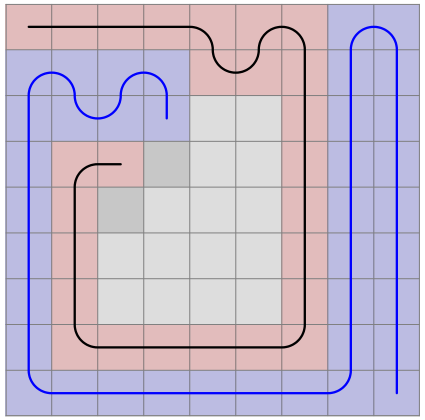
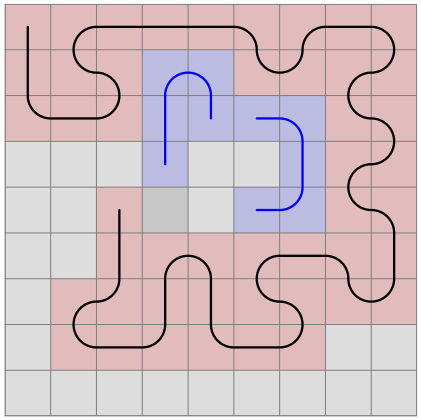
Similarly to the straight rules, these will only apply if there is no wall 2 distance to the left or right. Let's construct these preconditions.



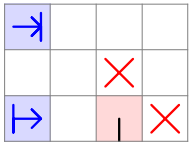
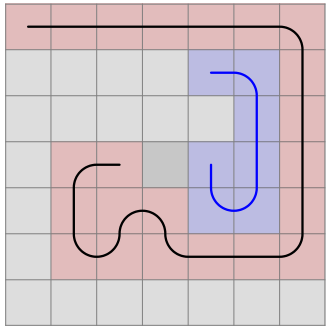
We are not finished. Did you notice the example above is not covered by these rules? We have to move the taken fields 1 and 2 steps to the side, both in straight and side direction.

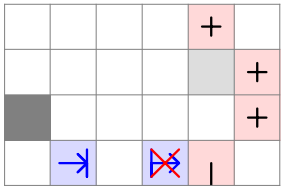
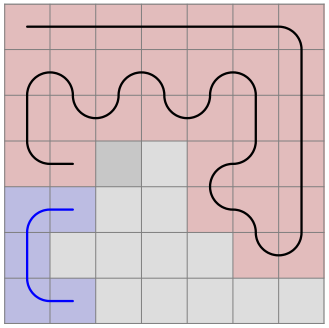
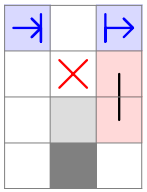
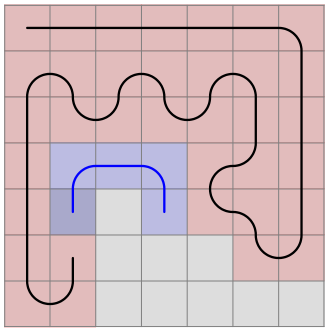
When any of the straight 2-distance rules are present, we don't need to check the side rules or the area created with the border. This is not entirely proven, but take these 9 x 9 examples:

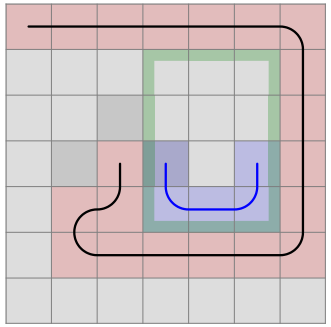



And these are the rest of the rules:

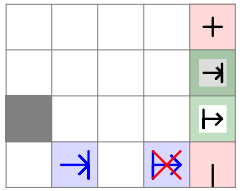
- This is what I started the 7 x 7 introduction with. I will call it Future L.



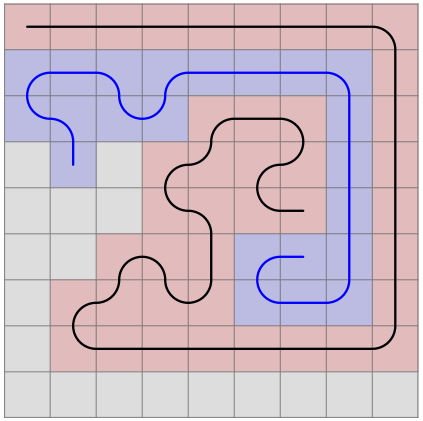
It would be a mistake to disable the right field.

So we need to check if an enclosed has been created on that side, but counting the area is unnecessary. Nevertheless, we can represent the rule this way, setting the circle direction to right:

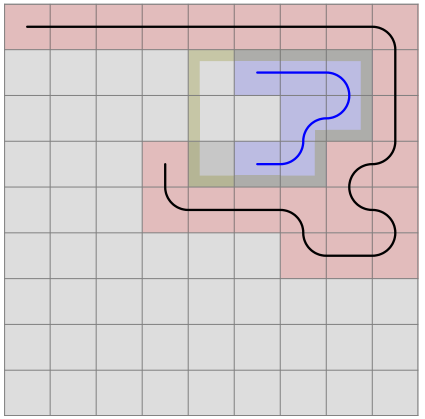


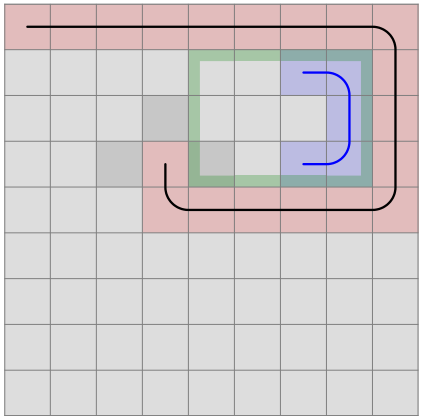
The code generator will examine if the count area start and end fields are 1 or 2 distance apart. In the first case, it will only determine in which direction the taken field straight ahead is going to, and if it is right, the forbidden field will take effect.
You may ask, why that field is "taken", not "taken or border". From what I found through some examples, if that field is border, the enclosed area on the right is impair, so the line cannot step in the other direction anyway. But it needs further examination.

To discover 9-specific patterns, I run the program keeping it left as long as the time to get to the first error is too big. After that, I will run it randomly. The first 13 826 walkthroughs are completed before we encounter a situation. It is similar to the last one we discovered on 7 x 7:

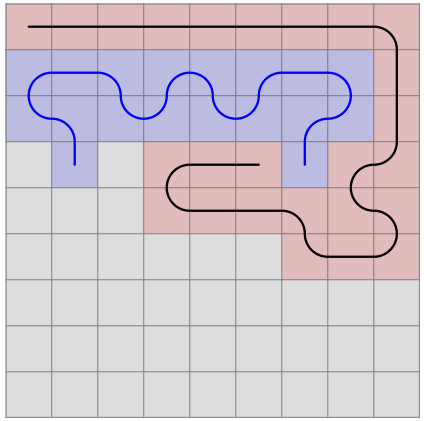


Let's simplify the pattern. Which will be impossible to fill?

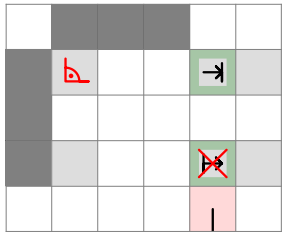




It is the picture on the left. Since the yellow-bordered area is impair, adding the (4,2) (4,3) (4,4) fields will be pair. We enter the area at (4,4), so we will exit at (4,3). Now we enter the 3 x 3 area in the top left corner at its side, (3,3) and will exit at (2,4). The results is two C-shapes on each side:



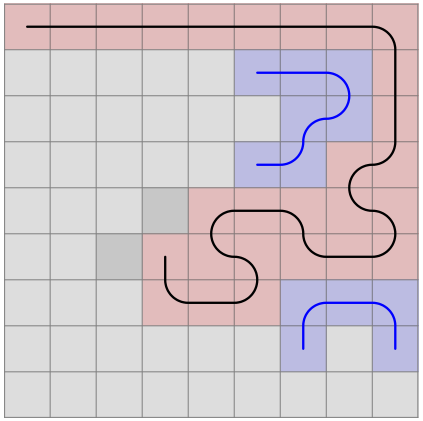
We can define a rule by marking the following fields and counting the area from the fields in front of the main line to the right:



Start_1 field is (4,3) and Start_2 field is (4,4) in the actual example. End field is (4,2). Direction of the circle: right (counter-clockwise). If the area is pair, we cannot step straight.

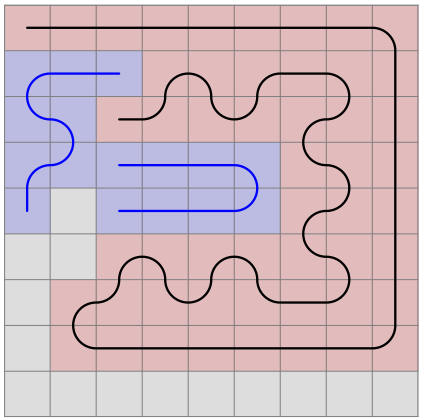
When generating code from the drawing, we have to check on which side the enclosed area was created. Here, we want it to be on the right side, so there are two cases to look at:
- The taken or border field beyond the end field is a taken field. In this case, if the field to its left is taken, its index must be lower. If the field to the right is taken, its index must be higher.
- It is the border. Add together the x- and y-coordinates to get a value. A higher value is closer to the end corner. Here, we compare the border field straight ahead and on its left, and we want the first-mentioned to be the smaller.

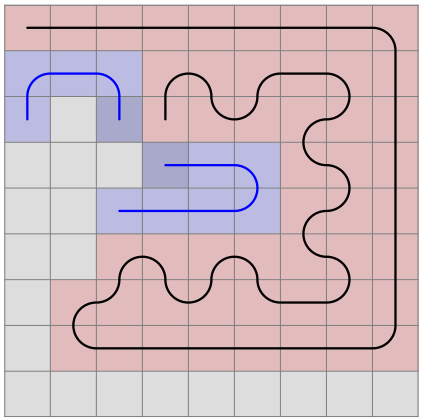
I have applied this rule rotated clockwise (besides mirroring it, of course), so that the live end can both come from the bottom and the right. But it can also come from the left in this example:



This will probably be another rule, because in this case it is not necessary to have an empty 3 x 3 field on the left.

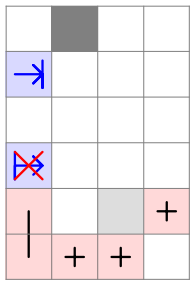
Now let's run the program further up to number 13 992:

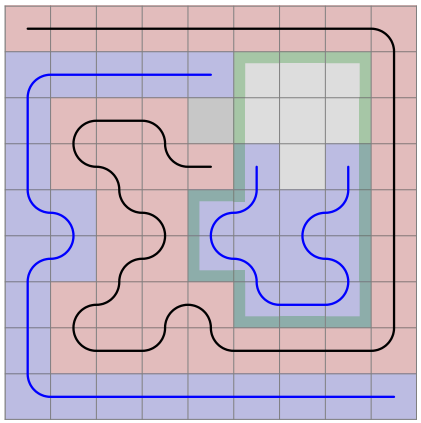




It is also just like the 7 x 7 rule, just with the extension of the area on the opposite side of the future line ends. But we can't simply remove the two taken fields on that side, because the line might continue in that direction, as it is the case here:

One certain situation reveals the incorrectness of the 7-rules when it comes to a 9-grid. In the following example, when I apply a rule rotated, it will disable a field that would otherwise be viable.

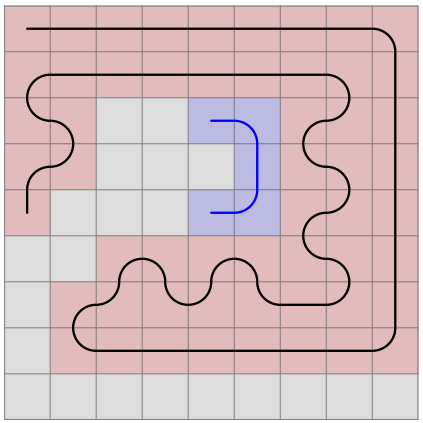




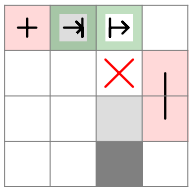
Rotating was not necessary to start with on 7 x 7, because no such situation occurred.

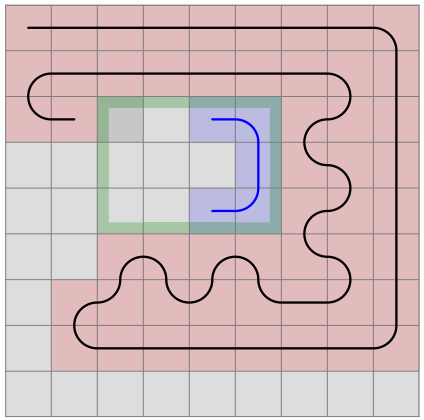
We can see that defining a rule with future line starts and ends does not tell us on which side the future line was created. That is the side that contains the enclosed area. We need to therefore replace such rules with area counting, which we actually already did, with the exception of Future L. Here the future line couldn't have been created on the other side, because that's the side the live end is at right now. And area counting is not always possible, like in this situation:

The next error, at 14 004 has something to with how I defined the universal rules of approaching an older section of the line, it needs to be reworked in light of the C-shape the main line can create with the border.

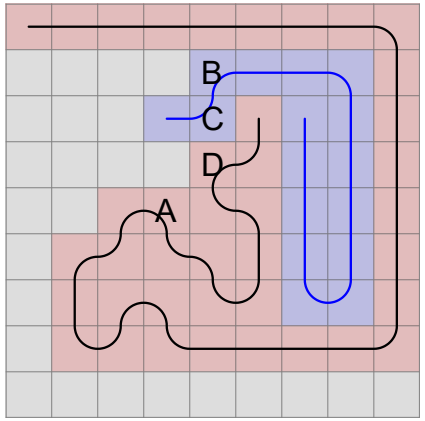


We need to take a few steps back, and then we can create the rule. It is similar to the universal 2-distance rule on the side, it just checks the field 2 behind and 1 to the side too. Even though the area counted is pair, now stepping to the right is disabled.

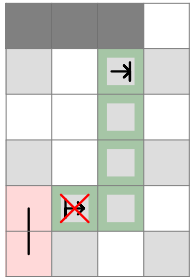


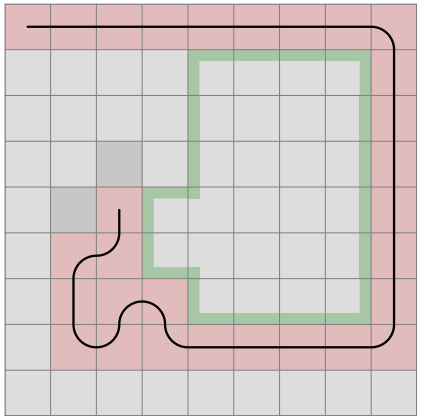


At 55 298, we get this:

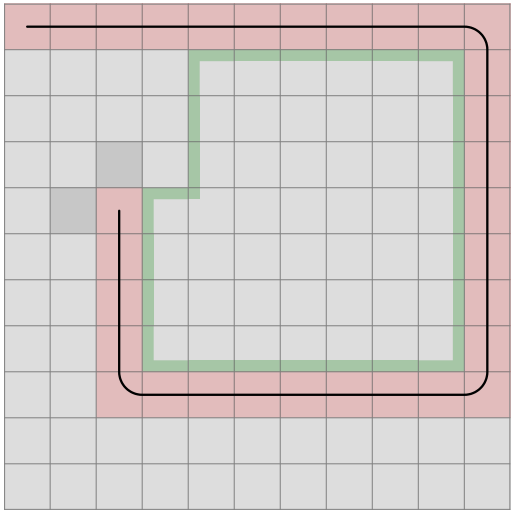


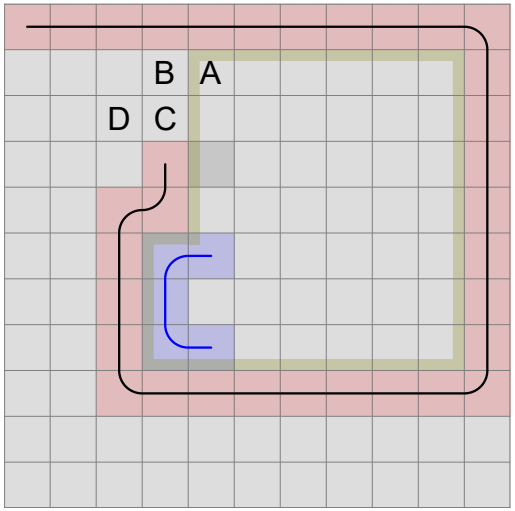
Let's analyze it! A double C-shape is created, because the line occupied the A field, and out of the B, C and D fields it exited the right-side area at C. It means, the area enclosed by the marked fields is pair. In this case, we shouldn't step right and the rule will therefore look like:





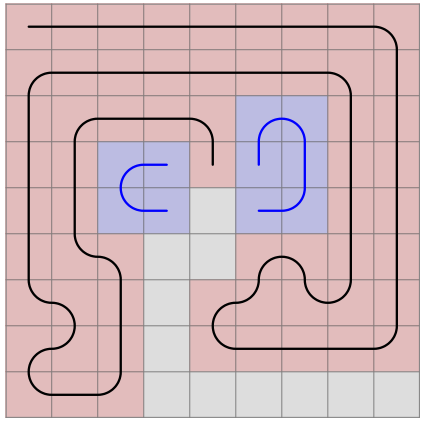
But what if from the A position, we step upwards in another situation? Compare these two on 11 x 11:

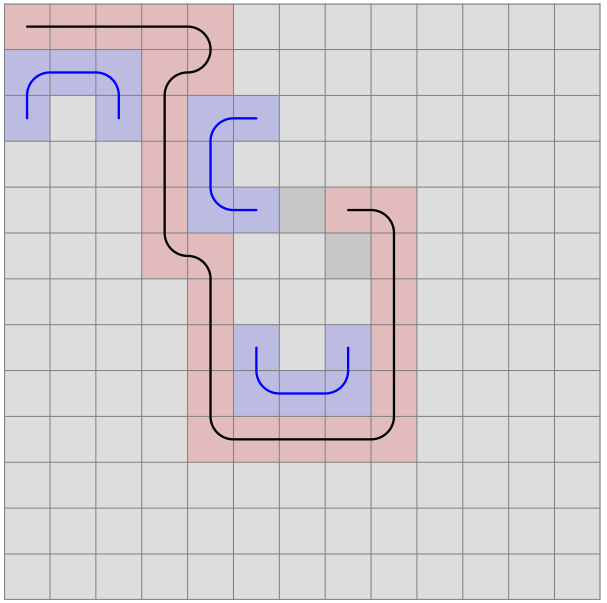




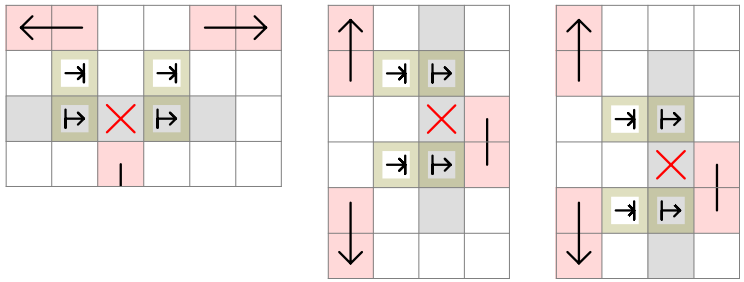
If the area we started with is pair, then the other will be impair. We can only enter the area at the light-gray field and will exit at A. From there we must go through B, C and D, and then a double C-shape is again created.

234 256 has at first sight something to do with future lines.



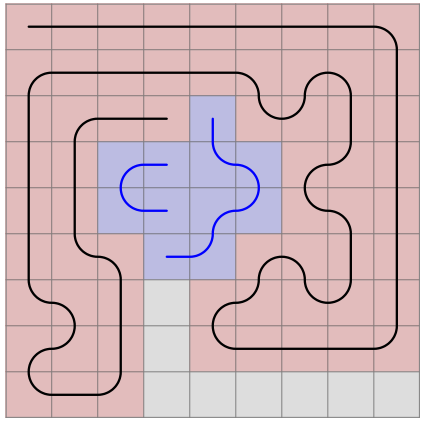


But it is more than that. Notice that enclosed areas has been created on both sides simultaneously. Because of the universal rules for approaching an older section of the line, now we have no option to move. The areas can be filled individually, but we cannot step to left and right at the same time.
We have to create 2-distance rules, which take both sides into account.

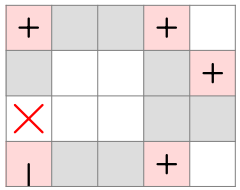


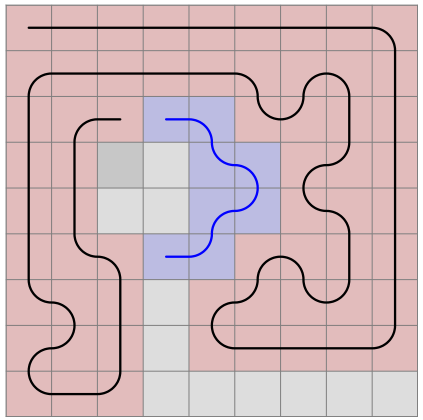
These are just a few of the possible combinations.
Any of the far straight rules (straight, mid across and across as I call them, depending on the horizontal distance of the obstacle) on the left side can be combined with any of those on the right side when the enclosed area is going to the same direction - left for left side and right for right side.
And the same is true when the pattern is rotated to the left or right side.
As far as porgramming concerned, it just needed a rework of the universal rules, we didn't need to make completely new ones.

As we run the program further, we will discover this at 227 200:

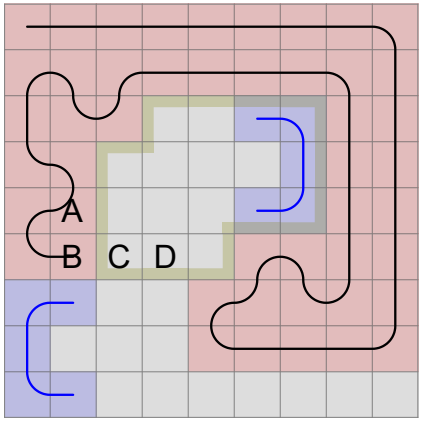


Intuitively, we can draw up the square, and let's mark the exit as well. There can be loops on the upper, lower and right side, they have no importance when tracing it back to the live end. There is only one way to go through.

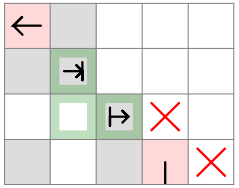


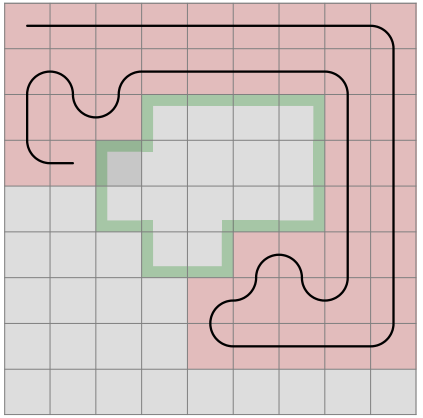


233 810 will look like:

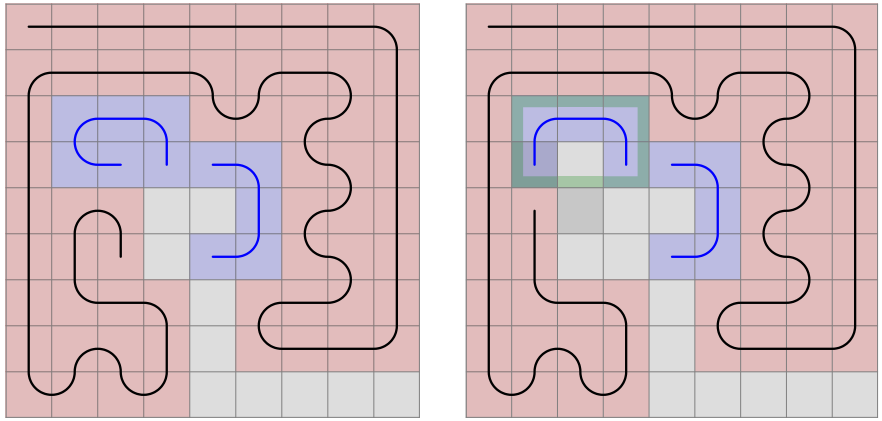


Once we step to A, it is unavoidable to get to B before entering the outlined area. It is because we can only reach B from the left or the bottom.
The area is impair, therefore we cannot complete it starting in C and ending in D.
If we omit the C field from the area, the area becomes pair. It is clear that the start and end field being across each other, a pair amount of fields cannot be filled. We must therefore enter the area now.

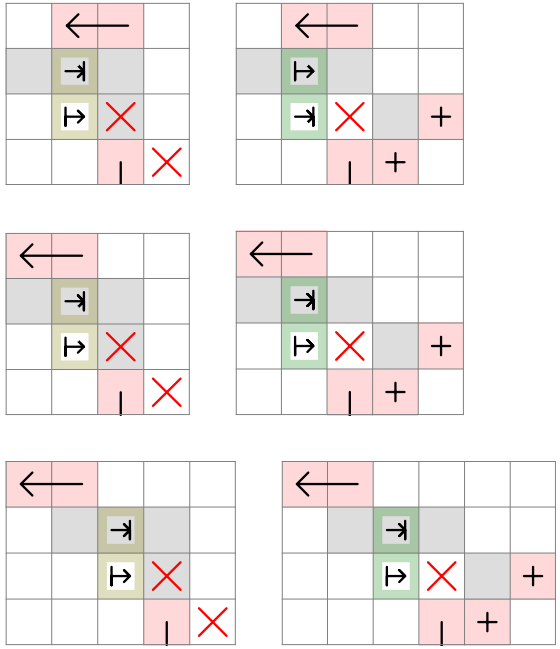




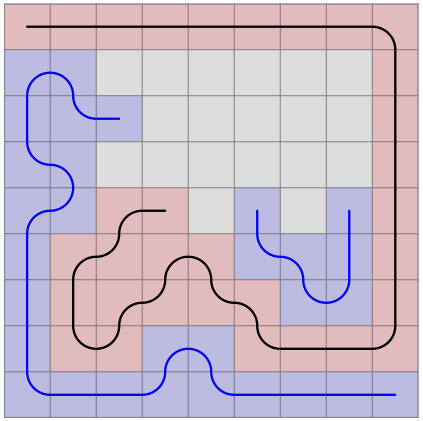
The same concept we encounter at 635 301, only the C-shape is created when we enter an area, on the other side of it.



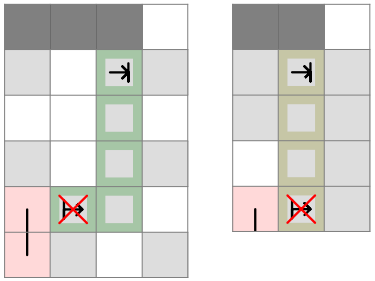
We have seen this in the third 9 x 9 rule. There the taken field next to the exit was in middle across position, and now it is across. And we also need to think about an obstacle straight ahead. Here are the original universal rules and their modifications.
Straight, circle direction left:



At 349 215, we find this:

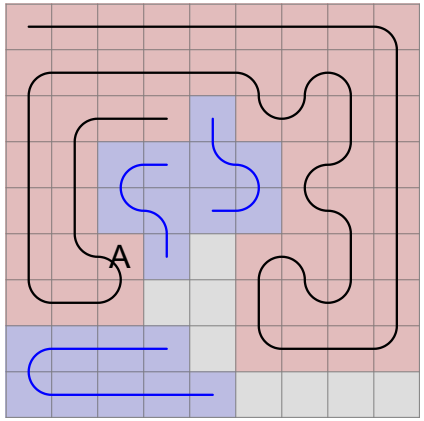


Though a double C-shape has been created in backwards direction, it indicates that the area on the right cannot be filled either.
We have made a similar rule previously. Now we need to simplify it.

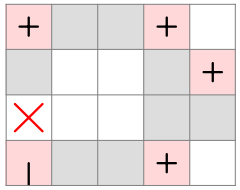


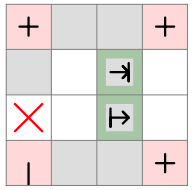
The area now has to be impair for the right direction to be forbidden. Essentially, we just added the three extra fields to the pair area.

478 361 is similar to what we have seen before, only now there is a 2-wide path to exit the area:



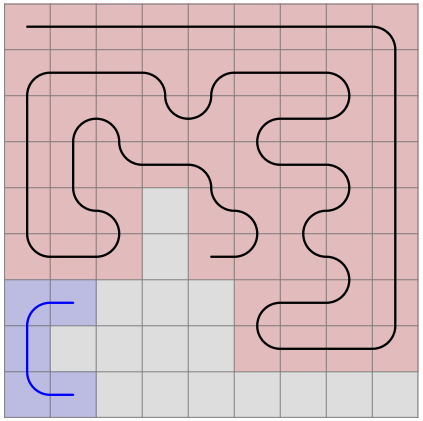
We have to mark where the area has been created in another way.

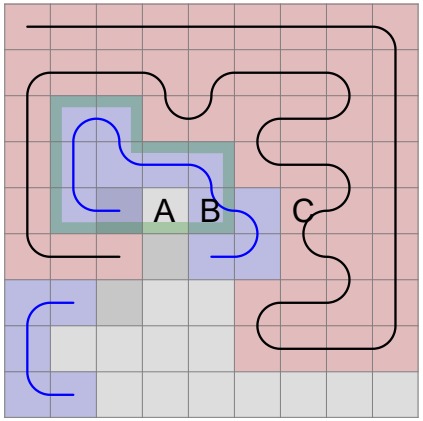




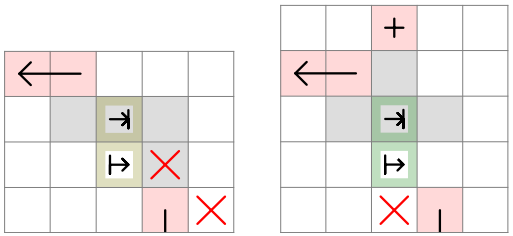
The taken field in the upper right corner is now checked for direction, but it is not enough. It can go upwards, and the exit of the area can still be on the bottom edge, just look at the example and imagine the live end was at A with the pattern already drawn. (On 11 x 11, it is possible to draw it.)
In order to establish an enclosed area, we must not encounter the bottom-right corner of the grid when walking along the edge of it.

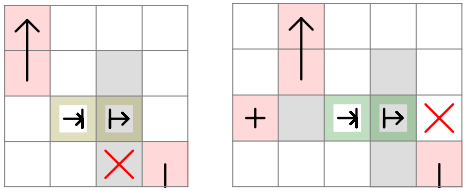
626 071 is:



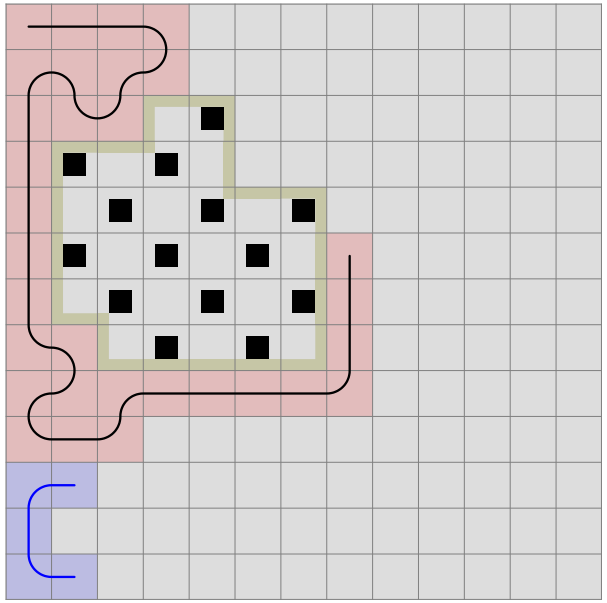


With the marked area being pair, if we enter the area by stepping left, we will exit at A. But we can only get there from B; if we entered from the top, nothing would fill B, and we cannot enter and exit it after we left the area - subtracting 1 from the area would make it impair, so then we couldn't have exited at A.
The taken field C creates a C-shape, which we need to step into from B.
The universal far across rule have to be extended. By default, we disable the option to step straight or right if the counted area is impair. When it is pair, we need to disable the left field.



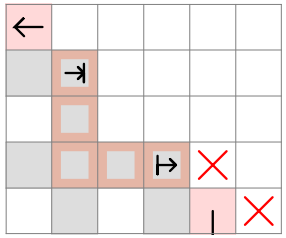


Let's mark the original example as a checkerboard.

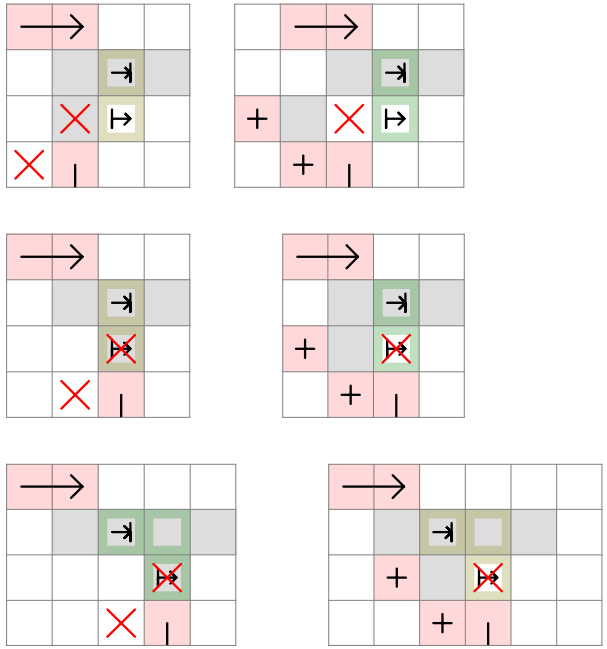


We enter at a black field and exit at black too, so the number of black fields should be one more than the number of white fields.
Here there are 14 black fields and 15 white. That's why the area cannot be filled. The up and right directions need to be disabled, so we can only step left.

This is the rule representation. The reddish arealine now means the arealine is impair, and we know that the entry and exit points are the arealine start and end fields.



Circle direction right:



Side, with taken fields above and below: