Rassemblement d'agents mobiles

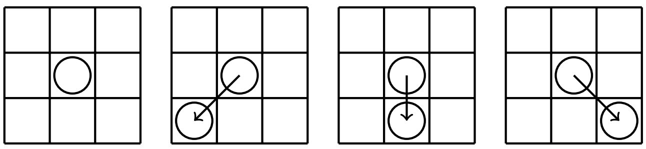
Éloi Perdereau

17 juin 2014

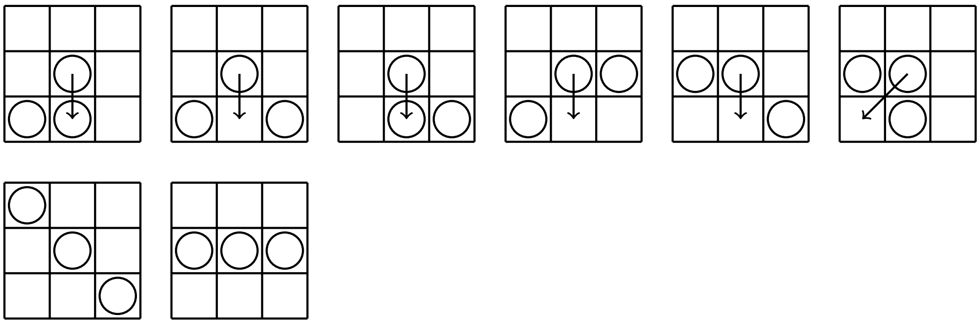
1 Cas

On omet les cas symétriques par rapport au robot du milieu (rotations de 90°, 180° et 270°.)

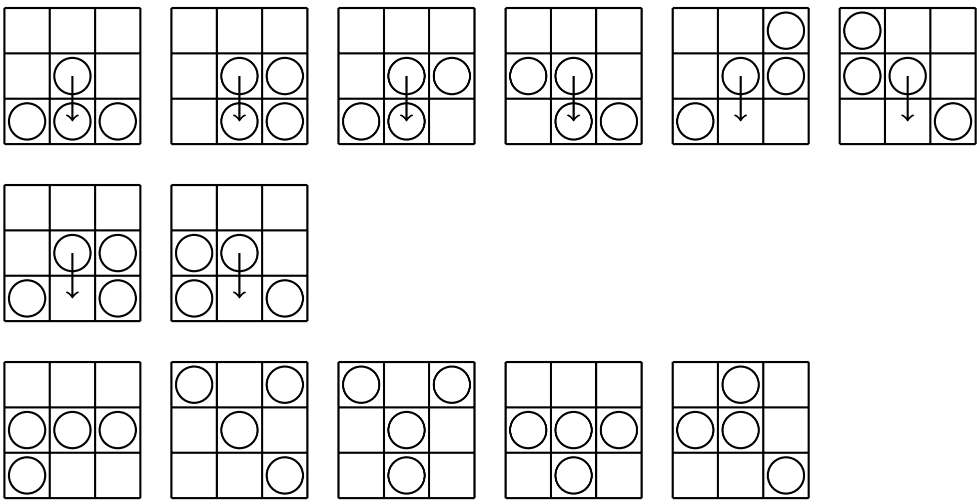
1.1 0 ou 1 voisins



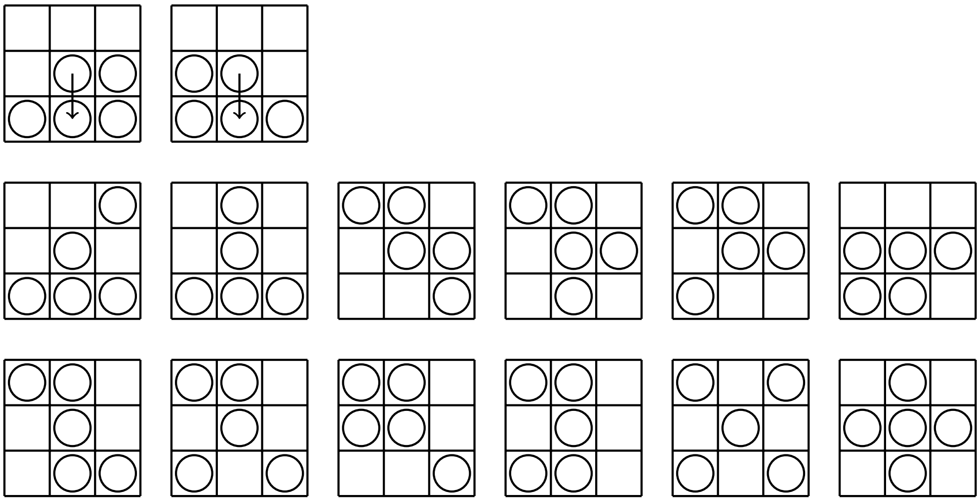
1.2 2 voisins



1.3 3 voisins



1.4 4 voisins



1.5 5 voisins et plus

Pour énumérer les cas avec 5 et 6 robots voisins, il suffit de prendre le complémentaire des cas avec respectivement 3 et 2 robots voisins. Aucun de ces cas n'entraine un mouvement de la part du robot concerné.

1.6 Extension

Tel quels, les cas 6 et 13 peuvent conduire à une déconnexion de l'espace. Pour y remédier, il faut que chaque robot mémorise son entourage d'une ronde sur l'autre (il ne retient que son entourage précédent.) Puis, si à la ronde précédente, il était dans le cas 6 ou 13, il faut qu'il vérifie si au moins une des cases suivante contient un robot : à droite, en bas et en bas à droite. Si ce n'est pas le cas, il revient à sa position précédente. Les cas symétriques sont définis de façon analogues.

Après avoir réglé ces cas de déconnexions, un autre problème survient avec les cas 5 et 6. Il se peut que l'espace alterne entre deux états ce qui rend le rassemblement impossible. Le problème vient du fait que des robots disposés en quinconce soient de nouveau en quinconce à la ronde suivante (avec des positions inversés.) Et ainsi, revenir à la position qu'ils occupaient deux rondes plus tôt. Pour y remédier, on va de nouveau utiliser l'entourage de la ronde précédente : Si à la ronde précédente, un robot était dans le cas 5 ou 6, et qu'il est désormais dans le cas opposé, alors il ne bouge pas pour cette ronde.

1.7 Formalisation

Algorithme 1 :

```
finish ← False;
k ← 0;
while not finish do
    N_k ← get_neighbors();
    if k%4 = 0 then
        if N_k is case 1.1.1 or
        (N_k is case 1.1.{2,3,4} and N_{k−2} = rotate180(N_k)) or
        (N_k is case 1.3.2 and N_{k−2} = rotate90(N_k)) then
            finish ← True;
        else if (i, j − 2) and (i + 1, j − 2) are both empty or both full then
            move according to N_k;
        end
    else
        if (N_{k−1} is case 1.2.{4,5} or 1.3.{5,6}) and ((i, j + 1), (i − 1, j), (i − 1, j + 1) are all
        empty) then
            move to (i − 1, j);
        end
    end
    k ← k + 1;
end
```

2 A single robot on the topmost row

We denote by $r(t)$ the single robot in the topmost row of the bounding box at step t. If there are more than one robot in the topmost row, $r(t)$ is not defined. The row and column of the cell occupied by $r(t)$ would be denoted $Y(t)$ and $X(t)$ respectively. In general we will assume $Y(t) = 0$ unless otherwise stated. The global configuration of robots at time t would be denoted by $C(t)$. If $C(t)$ satisfies the terminating conditions of the algorithm then it is called a GATHERED configuration.

Proposition 2.1. *If $r(t)$ exists and is on $(0, i)$, then there was a robot on cell $(0, i - 1)$, $(0, i)$ or $(0, i + 1)$ (or on cell $(-1, i - 1)$, $(-1, i)$ or $(-1, i + 1)$) at step $t - 1$.*

Proposition 2.2. *At step t, if a robot w was not in the neighborhood of $r(t)$, then robot w cannot be on the topmost row in step $(t + 1)$.*

The above properties imply that studying the neighborhood of $r(t)$ for new robots on the topmost row is sufficient to determine if the topmost row moved down.

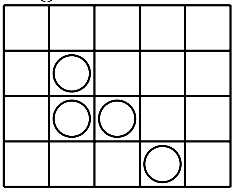
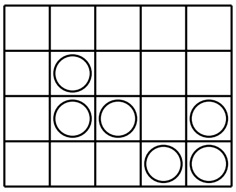
Lemma 2.3. *If $r(t)$ exists and the current configuration $C(t)$ is not* GATHERED *then there exists a constant c such that after c steps, either $BB(t + c) \subset BB(t)$ (i.e. the topmost row moves down) or a* GATHERED *configuration is reached.*

Démonstration. We define the graph $G_{single}(V_{single}, E_{single})$ as follows :
- V_{single} : neighborhood cases of $r(t)$
- $(u, v) \in E_{single}$ if u is the neighborhood of $r(t)$ and v is the neighborhood of $r(t + 1)$ such that $Y(t) = Y(t + 1)$ and the configuration $C(t + 1)$ is not GATHERED.

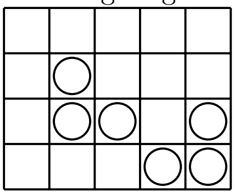
The graph generated by considering all possible single step transformations is shown on figure 1. If a node has no outgoing edges then in the next step $(t + 1)$ either there are no robots on row $Y(t)$ or a GATHERED configuration is reached. So the sink nodes in the graph satisfy the lemma. We notice multiple cycles in the graph ; if any cyclic path is followed by the algorithm then the topmost row might never move down. However, edges of the graph only represent single step transformations. We will show that the cyclic paths are not followed by the algorithm by studying 3 paths in the graph and this would be sufficient to prove that the lemma holds.

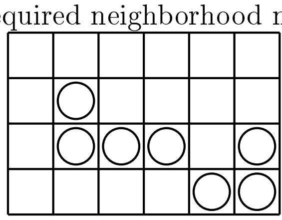
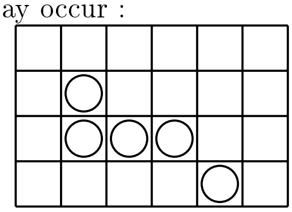
We denote by A to G the 7 nodes of G_{single} from top to bottom and left to right.
A transformation corresponding to an edge in G_{single} is called a *Left-move*, *Mid-move*, or *Right-move*, if $X(t) > X(t+1)$, $X(t) = X(t+1)$, or $X(t) < X(t+1)$ respectively.
Let t be the step at which the algorithm reaches the first node of the considered path, i.e. the neighborhood of $r(t)$ is represented by this node. For each path, the first edge is a *Right-move*.
Every neighborhood requirements stated below have been programmatically tested.

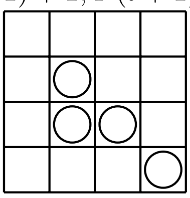
1. Consider the path $(B \to C \to D)$. For the algorithm to follow edge $B \to C$, one of the following neighborhood is required :

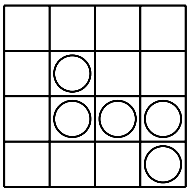
   According to the rule (1.3.5), the robot at $(X(t)+1, Y(t)+1)$ will move down and occupy $(X(t+1), Y(t+1)+2)$.
   Yet, for the algorithm to follow edge $A \to B$ at (t+1), the cell $(X(t+1), Y(t+1)+2)$ must be empty.
   Thus, the path $(B \to C \to D)$ cannot exist in any execution of the algorithm. Analogously, the path $(D \to A \to B)$ cannot exist either.
2. Consider the path $(B \to E \to D)$. For the algorithm to follow edge $B \to E$ we see that the following neighborhood is required :

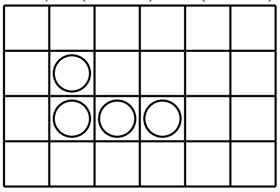
   According to the rule (1.3.6), the robot at $(X(t)+1, Y(t)+1)$ will move down and occupy $(X(t+1), Y(t+1)+2)$.
   Yet, for the algorithm to follow edge $E \to D$ at (t+1), the cell $(X(t+1), Y(t+1)+2)$ must be empty.
   Thus, the path $(B \to E \to D)$ cannot exist in any execution of the algorithm. Analogously, the path $(D \to E \to B)$ cannot exist either; neither does the paths $(B \to E \to B)$ and $(D \to E \to D)$ due to the symmetry of the nodes B and D.
3. Consider the path $(B \to D \to B)$. For the algorithm to follow edge $B \to D$ we see that different required neighborhood may occur :
   -  

     According to rule (1.2.5), the robot at $(X(t)+2, Y(t)+1)$ will move down and occupy $(X(t+1)+1, Y(t+1)+2)$.
   - 

     The robot $(X(t)+2, Y(t)+2)$ will either not move or move at $(X(t+1)+1, Y(t+1)+1)$, $(X(t+1), Y(t+1)+2)$, or $(X(t+1), Y(t+1)+1)$. In the latter case, a GATHERED configuration is reached at $(t+2)$.

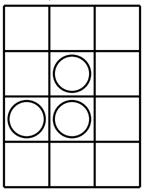
– 

The robot at $(X(t)+2, Y(t)+1)$ will either not move or move at $(X(t+1)+1, Y(t+1)+2)$ or $(X(t+1), Y(t+1)+2)$.

– 

A GATHERED configuration is reached at $(t+2)$.

Yet, for the algorithm to follow edge $D \to B$ at (t+1), the following neighborhood is required



Thus, the path $(B \to D \to B)$ cannot exist in any execution of the algorithm. Analogously, the path $(D \to B \to D)$ cannot exist either.

Due to the above arguments, all cyclic paths can be removed from the graph G_{single}. This directly implies that the statement of the lemma holds. □

The results of this section show that if there is only one robot in topmost (or bottom-most) row or equivalently if there is only one robot in leftmost (or rightmost) column, then the bounding box shrinks within a constant number of steps.

3 Multiple robots on the topmost row (draft !)

We denote by $l(t)$ the leftmost robot in the topmost row of the bounding box at step t. The row and column of the cell occupied by $l(t)$ would be denoted similarly to $r(t)$ defined in the previous section.

Proposition 3.1. *The robot $l(t)$ cannot successively perform* Mid-move*s indefinitely.*

Démonstration. We define the graph $G_{leftmost\_mid}(V_{leftmost\_mid}, E_{leftmost\_mid})$ as follows :

- $V_{leftmost\_mid}$: neighborhood cases of $l(t)$.
- $(u, v) \in E_{leftmost\_mid}$ if u is the neighborhood of $l(t)$ and v is the neighborhood of $l(t+1)$ such that $X(t) = X(t+1)$ and $Y(t) = Y(t+1)$ and the configuration $C(t+1)$ is not GATHERED.

We notice multiple cycles in the graph ; if any cyclic path is followed by the algorithm then the leftmost robot in the topmost row might never move. However, edges of the graph only represent single step transformations. We will show that the cyclic paths are not followed by the algorithm by studying 3 paths in the graph and this would be sufficient to prove that the proposition holds.

TODO (c'est ok sur papier)

□

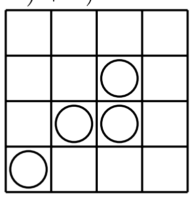
Proposition 3.2. *If $l(t)$ performs two* Left-move*s consecutively, than $X(l(t+3)) \geq X(l(t))$ or $C(t+4)$ is* GATHERED*.*

Démonstration. We define the graph $G_{leftmost\_left}(V_{leftmost\_left}, E_{leftmost\_left})$ similarly to $G_{leftmost\_mid}$ with $X(t) = X(t+1) - 1$. And the graph $G_{leftmost\_right}(V_{leftmost\_right}, E_{leftmost\_right})$ similarly to $G_{leftmost\_left}$ with $X(t) = X(t+1) + 1$.

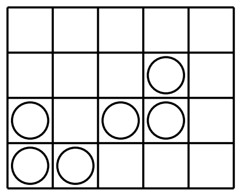
We will show that after $l(t)$ have performed two *Left-moves* consecutively, a *Right-move* or a *Right-move* is impossible.

We denote by A to H the 8 nodes of $G_{leftmost\_left}$ from top to bottom and left to right which is source or destination of any edge.

Let $l(t)$ be the robot performing two *Left-moves* consecutively.
The robot $l(t+2)$ can only be on cases E, F, G or H.
And cell $(X(t), Y(t))$ is empty because $NE(t+1)$, $E(t+1)$ and $SE(t+1)$ are empty.

- $l(t+2)$ is on case F or H. To perform a *Right-move*, $(X(t), Y(t))$ must be full for $l(t+2)$.
- $l(t+2)$ is on case E. To perform a *Right-move*, either $EE(t+2)$ must be occupied or $S(t+2)$ must be on case (1.2.6). The former case is impossible because $EE(t+2)$ is $(X(t), Y(t))$ which is empty. Moreover, to follow edge $(D \to E)$, at least one of $SWW(t+1)$ and $SSWW(t+1)$ must be occupied. So at $t+2$, either $SSW(t+1)$ or $SSWW(t+1)$ are occupied which prevent $l(t+2)$ to perform a *Right-move* ; or $C(t+3)$ is GATHERED.
- $l(t+2)$ is on case G and $(X(t+2), Y(t+2)+2)$ is full because to follow edge $(D \to G)$, $l(t+1)$ must have the following neighborhood : 
  Thus, to perform a *Right-move* from case G, either $(X(t+2), Y(t+2)+2)$ must be empty or both $(X(t+2)+2, 0)$ and $(X(t+2)+2, 1)$ must be full. Or the configuration is GATHERED.

So after two *Left-moves*, a *Right-move* is impossible (or the configuration is GATHERED.

According to $G_{leftmost\_mid}$, cases E, G and H cannot lead to a *Left-move*. Moreover, for $l(t+2)$ to be in case F, $l(t+1)$ must be in the following neighborhood :  In this case, $(X(t+1)-1, Y(t+1)+1)$ will move down and $(X(t+2), Y(t+2)+2)$ will be occupied. But to perform a *Right-move* from case F, this cell must be empty.

So after two *Left-moves*, a *Right-move* is impossible (or the configuration is GATHERED.

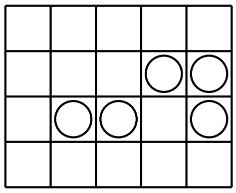
□

Proposition 3.3. *Two consecutive* Left-moves *starting at column $X(t)$ cannot occurs twice for the same column.*

Démonstration. Let t be the starting time of the two consecutive *Left-moves*.
The robot $l(t)$ have a single neighbor : $SW(t)$.
At $t+2$, $(X(t+2)+2, 0)$ is empty. Thus, to repeat two *Left-moves*, $(X(t+2)+1, 0)$ must perform a *Left-move* and be in case A, i.e the same neighborhood than $l(t)$ (after eventually a few *Right-moves*). But in $G_{leftmost\_mid}$, there are no path from any destination of edges in $E_{leftmost\_left}$ to case A. □

Proposition 3.4. *A* Left-Right-move *starting at column $X(t)$ cannot occurs twice for the same column.*

Démonstration. We showed above that a *Left-Right-move* starting at case D is impossible or a GATHERED configuration is reached after a constant number of steps. A *Left-move* starting at A leads to cases that cannot perform a *Right-move* according to $G_{leftmost\_right}$. There are edges $C \to E$ and $C \to F$ left.

To follow edge $C \to E$, $l(t)$ must have the following neighborhood :  So $l(t+1)$ be in case (1.2.3) and perform a *Right-move*. The robot $l(t+2)$ cannot be in case (1.2.1) (see proof above) and cells $(X(t+2)-2, 1)$ and $(X(t+2)-2, 1)$ are empty. Thus, from case A or C, $(X(t+2)-2, 1)$ must be occupied.

To follow edge $C \to F$, there are 23800 possible cases and I don't know if a *Right-move* at $t+1$ can lead to a *Left-move* case.
TODO □

Proposition 3.5. *After $l(t)$ performs a* Left-move*, if $l(t+1)$ doesn't, than it will eventually perform a* Right-move*, or a* GATHERED *configuration will be gathered or there will be only one robot in the topmost row. i.e $L(M)^+L$ is impossible.*

Démonstration. The only possible path for $l(t)$ to perform $L(M)^+L$ is through edges $C \to F$ or $D \to F$, than $l(t+1)$ perform a *Mid-move* to case D.

- Edge $D \to F$: After follow this edge, $SS(t+1)$ is occupied (see item 2 of the proof of lemma 2.3), thus $l(t+1)$ cannot perform any *Mid-moves*.
- Edge $C \to F$: $E(t)$ and $SE(t)$ may move, but in every cases, they will occupy either $EE(t+1)$ or $SEE(t+1)$ or both. But for $l(t+1)$ to perform a *Mid-move* to D, $SE(t+1)$ must have only one neighbor : $l(t+1)$. Thus, $l(t+1)$ cannot perform the proper *Mid-move*.

□

Proposition 3.6. *After a* Left-move*, a series of* Mid-moves *starting at case G, and a* Right-move*, a* Left-move *is impossible.*

Démonstration. TODO
is this prop even true ? □

If all theses propositions are correct, we can say that $l(t) < l(t+c)$ after a constant number of steps, or a configuration is GATHERED or there is only one robot in the topmost row.

A Single robot movements

B Leftmost robot *Mid-moves*

C Leftmost robot *Left-moves*

D Leftmost robot *Right-moves*

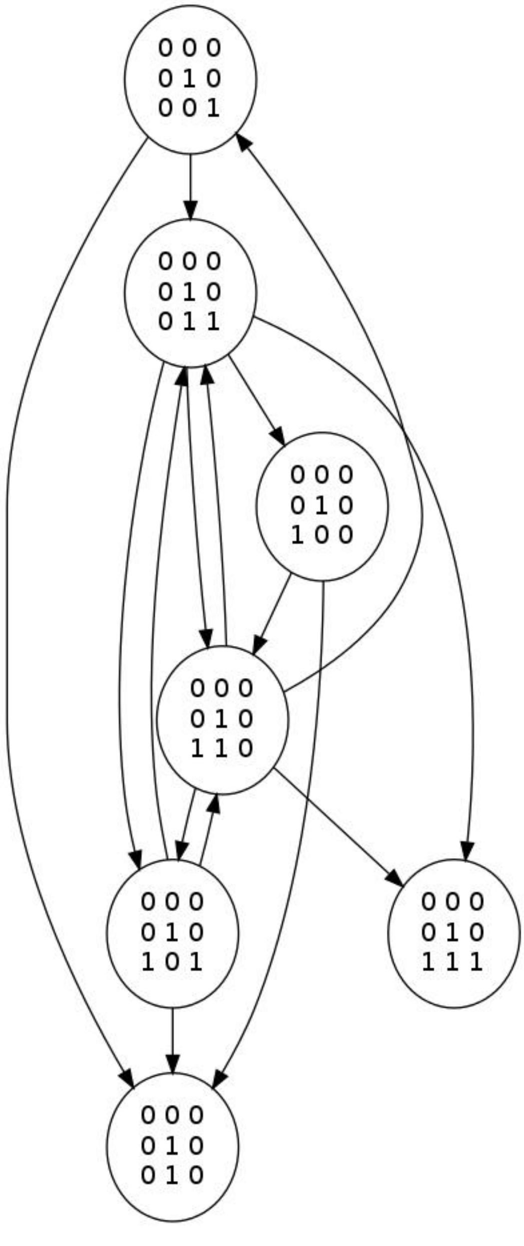


FIGURE 1 – Single robot

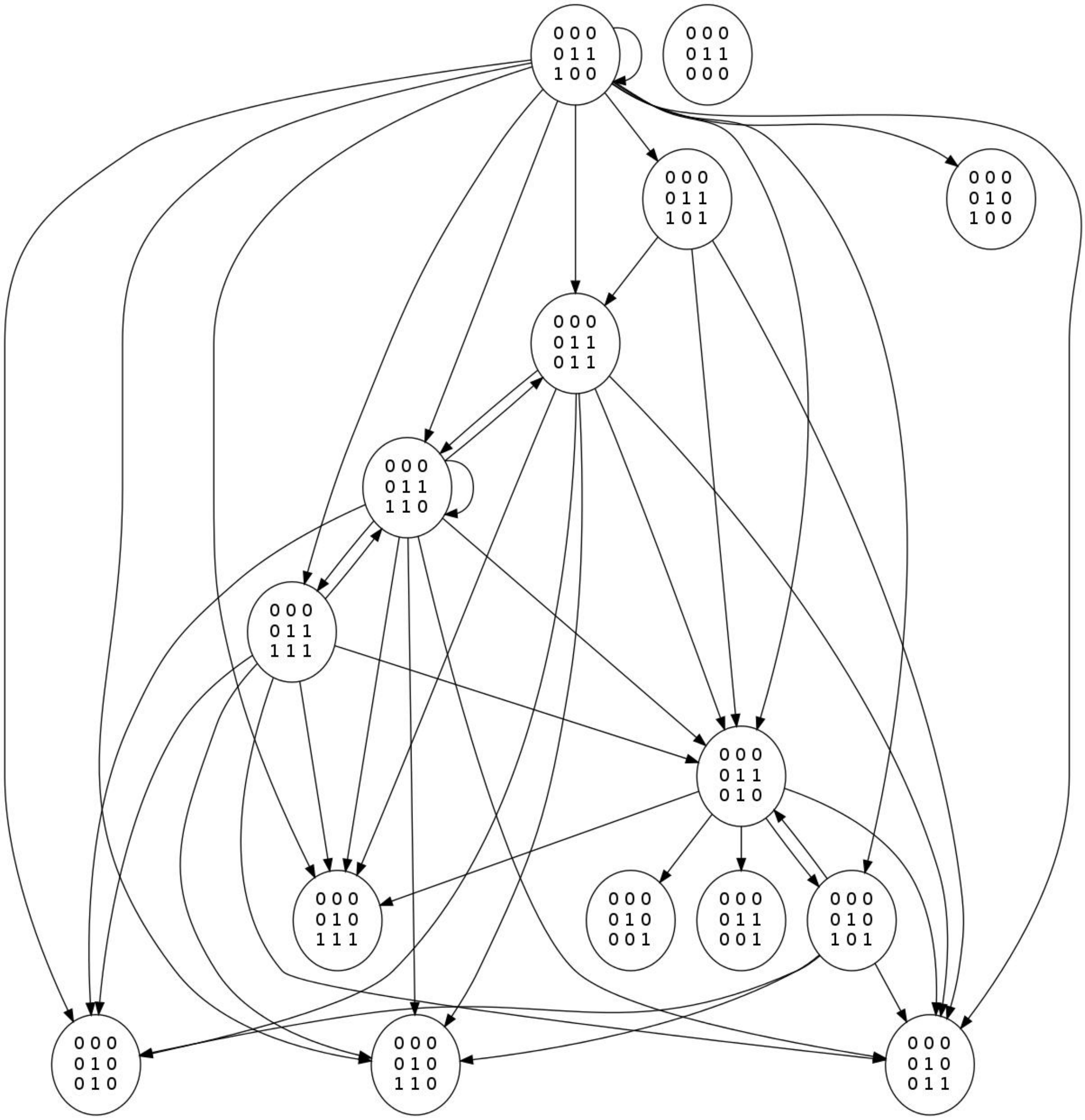


FIGURE 2 – *Mid-move*

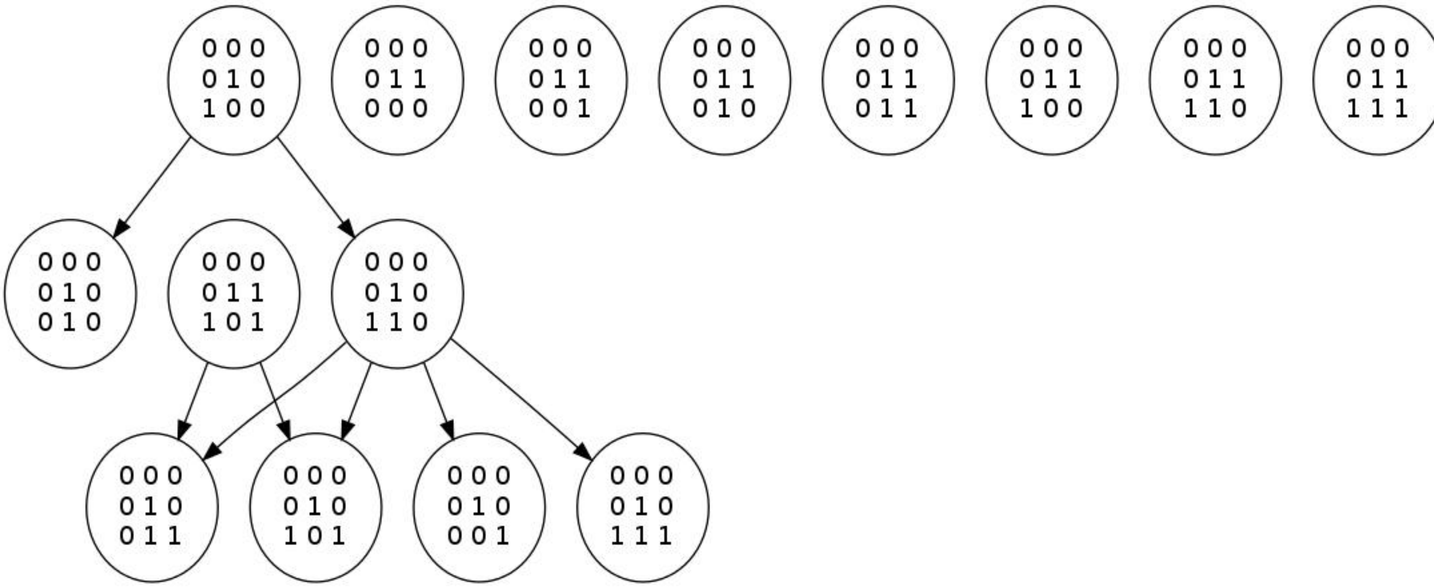
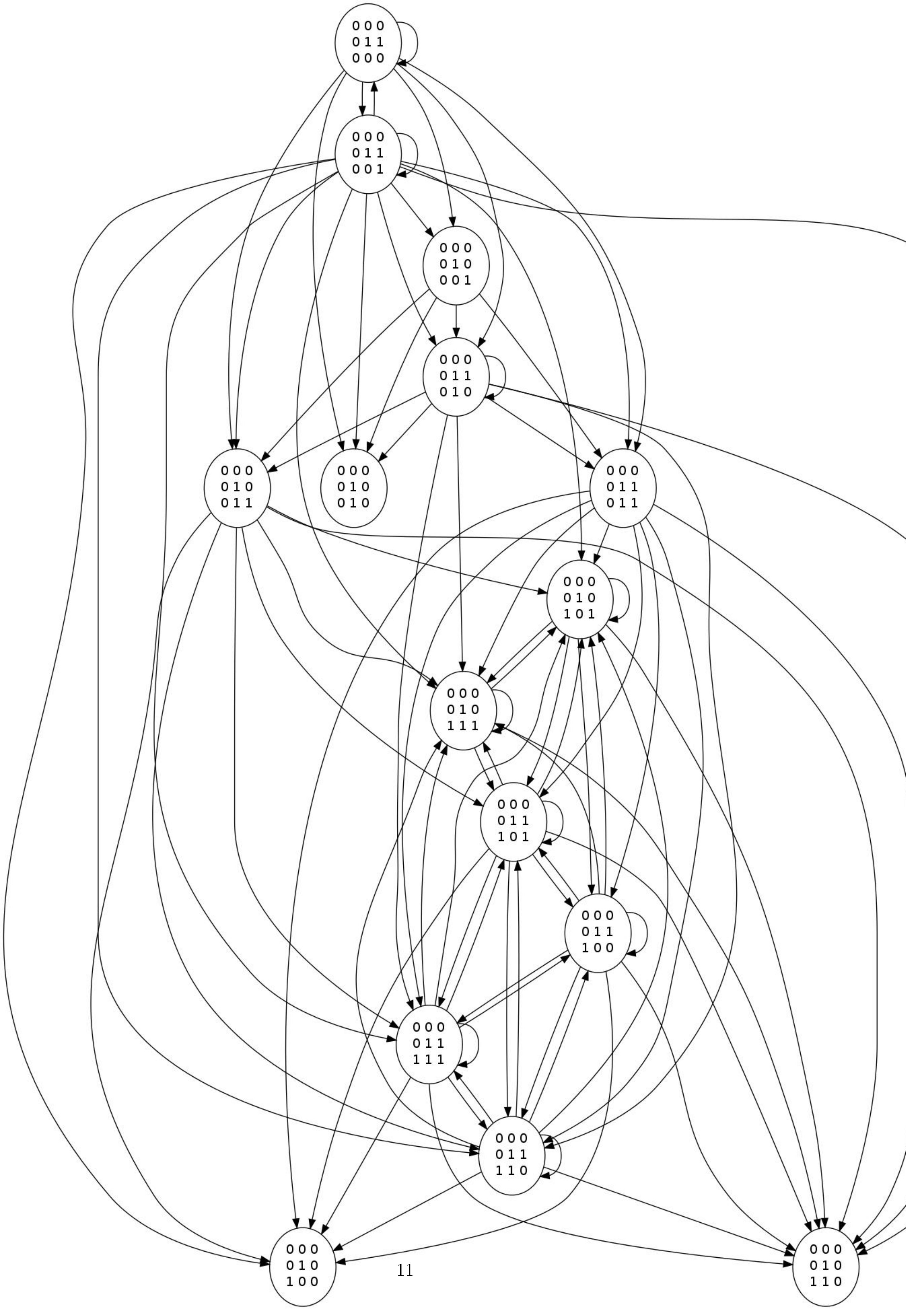


FIGURE 3 – *Left-move*


0 0 0
0 1 1
0 0 0
0 0 0
0 1 1
0 0 1
0 0 0
0 1 0
0 0 1
0 0 0
0 1 1
0 1 0
0 0 0
0 1 0
0 1 1
0 0 0
0 1 0
0 1 0
0 0 0
0 1 1
0 1 1
0 0 0
0 1 0
1 0 1
0 0 0
0 1 0
1 1 1
0 0 0
0 1 1
1 0 1
0 0 0
0 1 1
1 0 0
0 0 0
0 1 1
1 1 1
0 0 0
0 1 1
1 1 0
0 0 0
0 1 0
1 0 0
0 0 0
0 1 0
1 1 0