Optimally Gathering Two Robots

Some smart people

April 2017

1 Introduction

2 Problem Definition

The gathering problem for two robots is achieved in a self-stabilizing fashion in finite time if and only if, for any starting position, phase and colour, after a finite number of activations, both robots are located at the same coordinates.

3 Model

We consider a system of two robots. Those robots are mobile in the plane, modelled as the $\mathbb{R}_2$ space.

We consider the following model for two robots : Both robots execute cycles of three phases : LOOK, COMPUTE AND MOVE. We assume the LOOK phase is instantaneous, and that the COMPUTE and MOVE phases have an unknown, but finite duration.
Robots are anonymous, meaning there are no fixed way of distinguishing them. However, for the sake of practicality, they are called A and B. Note that switching their name during the proof has no consequence on its correctness.
Both robots are also fitted with lights which can take one of two colours, called black and white.
During its LOOK phase, robot A takes a snapshot of both robot B's colour and position, relative to itself.
A robot is able to change the colour of its own light at the end of its COMPUTE phase, i.e. at the moment corresponding to the activation following its COMPUTE phase activation
If it is not executing the LCM cycle, a robot is WAITING. We consider a fair, asynchronous scheduler. At every instant, the scheduler can activate either A, B or both.

We call δ the minimum distance a robot can travel through a single MOVE phase if its target is farther than delta. If the target of the robot is closer than δ when the robot activates its MOVE phase, then we assume the robot reaches its target.

In other words, if a robot has a target at a distance x, we assume that, at the end of its move phase, the robot has moved a distance in $[min(\delta, x), x]$. The exact position reached is determined by the adversary scheduler.

4 Proposed Algorithm :

4.1 Problem

To solve the problem, a gathering algorithm needs to make sure of three things :

- If the robots are synchronized, they need to move towards the middle point
- If the robots are not synchronized, one needs to move towards the other
- In that case, the other robot must not move (only necessary in ASYNC)

4.2 Algorithm

We consider the following algorithm :

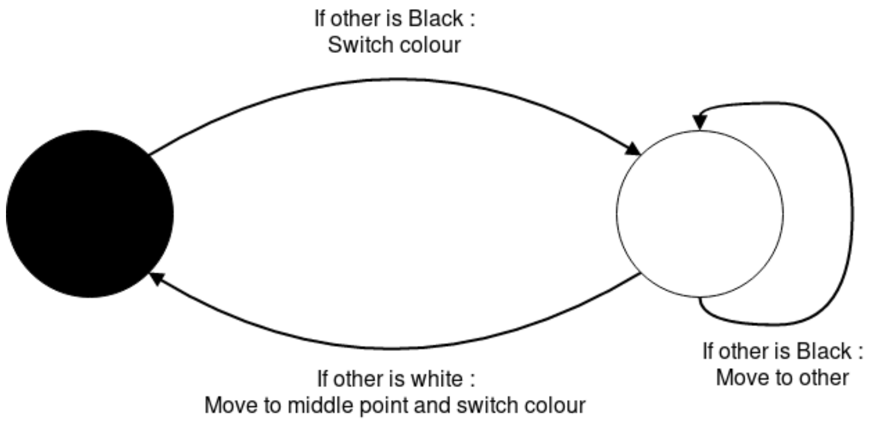

Figure 1: Algorithm

Theorem 1. *The algorithm solves the gathering problem for two robots in a self-stabilizing fashion in finite time for the ASYNC model.*

4.3 Principle

The principle of the algorithm is the following :

If both robots A and B start in the same colour and are fully synchronized, they move along the white to black and black to white vertexes. In that case, the each move towards the middle point.

If they get de-synchronized or start in different colours, the robot in black is forced to stay put while the robot in white moves towards it on the white to white vertex.

This algorithm is quite trivial to prove for both the FSYNC and SSYNC model. However, proving the algorithm for the ASYNC model requires us to take into account what exactly happens during the de-synchronisation.

5 Proving the algorithm :

5.1 Introduction

To prove this algorithm for the ASYNC model, we need to consider every possible configuration of the algorithm.
We consider the different types of phases that can be reached by the algorithm :

	W	C2H	C2O	M2O	W	C2W	C2B	M2H	C2W
W	🟩 A1	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛
C2H	🟨 A2	🟩 A9	⬛	⬛	⬛	⬛	⬛	⬛	⬛
C2O	🟩 A3	🟩 D1	🟩	⬛	⬛	⬛	⬛	⬛	⬛
M2O	🟩 A4	🟩 D2	🟩	🟩	⬛	⬛	⬛	⬛	⬛
W	🟩 B1	🟨	🟩 B3	🟩	🟩 A5	⬛	⬛	⬛	⬛
C2W	🟨 A6	🟩	🟩	🟩	🟨	🟩	⬛	⬛	⬛
C2B	🟩 B2	🟩	🟩	🟩	🟩 A7	🟩	🟩	⬛	⬛
M2H	🟩 C1	🟩 A11	🟩 C2	🟩 C3	🟩	🟩 A8	🟩	🟩 A10	⬛
C2N	🟩 F1	🟩 F2	🟩 F3	🟩 F4	🟩 F5	🟩 F6	🟩 F7	🟩 F8	🟩 F9

Figure 2: Configurations

- W : Wait
- C2H : Compute to Half (at the end of this compute phase, the robot switches its colour to black and moves towards the middle point)
- C2O : Compute to Other (at the end of this compute phase, the robot stays white and moves towards the other robot)
- M2O : Move to Other

- C2W : Compute to White (compute phase that leads to no motion)
- C2B : Compute to Black (compute phase that leads to no motion)
- M2H : Move to Half
- C2N : Compute to Nothing (compute phase that leads to no motion)

Since robots are anonymous, we only need to consider half of the possible configurations.
The named configurations (i.e. A1, F4 ...) will be used as targets in the configuration graphs.
The configurations with yellow backgrounds pose an interesting problem : In those cases, the simultaneous activation of A and B by the scheduler is not perfectly defined. Indeed, activating A then B or B then A does not lead to the same configuration. In those cases, we need to consider two different simultaneous activations.

We then divide the complete set of configurations in six subsets :

(a) SYM : configurations where A and B are in, or can reach, identical states.

(b) ASYM : configurations where A and B are differentiated into a black robot and a white robot and have no way of going back to being identical.

These two subsets cover the normal operation of the algorithm.

(c) FAULTY 1 : faulty configurations after a white to black de-synchronisation.

(d) FAULTY 2 : faulty configurations after a black to white de-synchronisation.

These two subsets cover faulty behaviour allowed by the algorithm.

(e) Illegal : configurations that cannot be reached by the algorithm but need to be taken into account as possible starting configurations to prove self-stabilization

(f) Gathered : Specific configurations that can only be reached if the gathering is complete.

Each subset is drawn as a graph of its configurations and the possible exits. Some entrances of FAULTY 1 and FAULTY 2 are also represented for better readability.
Each graph is divided in two parts : the top part, called nominal behaviour and a bottom part called terminal behaviour.
The behaviour of the algorithm is nominal when a robots are not gathered. Certain nodes, represented in the bottom part, will change their behaviour if the gathering is achieved during their activation. This is the case when a white robot activates its look phase and perceives the other robot and itself on the same coordinates.

5.2 Proof

To prove the correctness of the algorithm, we need to prove that :

- every nominal behaviour leads to a terminal phase
- every terminal phase leads to the gathering

5.2.1 ASYM

ASYM is the subset that is reached after a successful de-synchronization of A and B where A and B are now differentiated into a white and a black robot.

The easiest subset of the configuration to prove is the ASYM subset. Indeed, as it has no possible exits. We can easily prove that, once the system is in a ASYM configuration, fair activations of the robots will lead to :

- A moving towards B
- B not moving

in their nominal behaviour. It is therefore easily proven that the nominal behaviour leads to both robots sharing the same coordinates in a finite number of steps. Once it is done, the following activation of robot A leads to either the B1 or B2 configuration. The following activation of A then leads to either F5 or F7, which, in turn, either leads to B1 or B2. The system, is now stuck in a no-movement loop.

Therefore, the ASYM subset of configuration is a valid subset.

5.2.2 FAULTY 1

The FAULTY 1 subset is reached with a white to black de-synchronization. It is then possible to have robot A moving towards the middle point and B moving towards A.

The first thing to notice in the FAULTY 1 subset is that, in nominal behaviour, the system can only exit the subset by going to the ASYM subset, which we have proven to be a valid subset. It is also worth noticing that, while three cycles exist in this subset, none of them can actually be kept under the fair scheduler assumption, i.e. the require one robot to never be activated. And, while it is possible to switch from the first cycle to the second, and from the second to the third, breakage of the third cycle leads the system to the ASYM subset.

Therefore, the nominal behaviour of FAULTY 1 is valid.

Note that the only configuration of FAULTY 1 whose behaviour is modified by achieving the gathering is C1. C1 can then lead to either B1 and ASYM, or F5 and ASYM. Finally, if only the white robot is activated, it is possible to cycle between C1 and F8. However, the fair scheduler assumption requires this cycle to be broken and the system to move to the ASYM subset.

We can conclude that, despite being an abnormal subset, FAULTY 1 is merely a temporary subset, provided the scheduler is fair.

5.2.3 FAULTY 2

The FAULTY 2 subset is reached with a black to white de-synchronization. It is then possible to have robot A moving towards B and B targeting the middle point.

FAULTY 2 poses quite a problem : It is possible to enter FAULTY 2 from A6 in the SYM subset. However, it is also possible to exit FAULTY 2 to A2 in SYM. This A2 configuration contains an abnormal target. Therefore, it is possible for a scheduler to create a FAULTY 2 $< - >$ SYM cycle.

We need to prove that this particular cycle does not threaten the gathering. To do so, we notice that, in order to keep this cycle running, we need to go through the A6 configuration, and then the A2 configuration.

If we assume that, in the A6 configuration, we have :

- A, white, in position 0 with no target
- B, black, in position x with no target

Then it is possible to prove, by exploring every possible path, that when reaching A2 :

- A is in position $[min(\delta, x), x]$ with no target
- B is in position x with either $\frac{x}{2}$ or $[\frac{x}{2}, x]$ as a target.

Now, to go back to A6, the system needs to execute B's target. Also, A will target the middle point and execute this movement. This means that, when back to A6, the distance between A and B is now $[-\frac{x}{2}, x - 3\delta]$ which is less than x.

Thus, while it is an abnormal state, FAULTY 2 cannot prevent A and B from moving closer to each other. If the distance is inferior to δ, then at the end of D2, the gathering is achieved.

The nominal behaviour of FAULTY 2 is valid.

FAULTY 2 does not have a terminal behaviour. We can, however, note that in terminal behaviour, A2 cannot lead to A6 (proved in the SYM section). This means that the FAULTY cycle leads to the gathering, and is then broken when reaching A2.

5.2.4 SYM

The SYM subset is considered the normal starting starting subset (A and B waiting in white). It comprises the FSYNC states (A1 to A1 through simultaneous activations) and every state that can be reached by A1 and lead back to A1 (except through FAULTY 2).

In this subset, it is possible to exit through ASYM, FAULTY 1 FAULTY 2. It is also possible to cycle through SYM. This can either be done from A6 to A2, A7 to A5 or by cycling back to A1. In each case, cycling through SYM implies both robots targeting the middle point while not moving, and then both of them moving towards the middle point. Therefore, SYM nominal behaviour is valid.

Achieving the gathering implies terminal behaviour for nodes A1, A2, A3, A4 and A6.

- A1 can lead to F1 which leads to A1
- A1 can lead to F9 which leads to A1
- A2 can lead to ASYM

- A2 can lead to F2 or F8 which leads to A2, FAULTY 1 or ASYM
- A3 can lead to F3 or F4 which leads to A3, A4 or A1
- A3 can lead to A4
- A4 can lead to A1
- A4 can lead to F1 which leads to A1
- A4 can lead to F4 which leads to A1 or A4
- A6 can lead to A1
- A6 can lead to F1 which leads to A1
- A6 can lead to F6 which leads to A1 or A6

  For one, it is clear that it is no longer possible to keep the SYM $<->$ FAULTY 2 loop, as neither A6 nor A2 can lead to their faulty targets any longer. Furthermore, Reaching A2 with an invalid target and after achieving the gathering now will lead to FAULTY 1 or ASYM, which are valid subsets. Looping around to A2 would, indeed, require an unfair scheduler. It is also clear that, once the gathering is complete, SYM can either loop without motion or lead to a valid subset. Therefore, SYM is a valid subset.

5.2.5 Illegal

This subset covers configurations that are not possible to reach from normal behaviour. This is necessary to ensure self-stabilization. We can quickly notice that no cycle exists within the illegal subset and that this subset needs to exit to either SYM or FAULTY 1.

Therefore, this subset is valid.

5.2.6 Gathered

This subset has already been proven above.

5.2.7 Self-Stabilization

To prove the algorithm works in a self-stabilizing fashion, we need to prove that every possible starting configuration leads to a valid gathering. To do so, we need to prove two things :

- Starting from anywhere leads to gathering
- No false target can be kept indefinitely

This second hypotheses is necessary as we do not take the target into account in the configurations.

We already proved that every subset of configurations is valid and leads to the gathering.
To prove that no target can be kept infinitely in the general execution, we only need to prove it for SYM and ASYM, as all other subsets eventually lead to either of them.
This proof is quite trivial for ASYM, as it is a cycle of motion for the white robot.

For SYM, we notice that cycling through the subset requires moving from the leftmost configurations (A1-2-9) to the rightmost configurations. this implies crossing the yellow configurations and, again, provided the scheduler is fair, activating both robots through their MOVE phase once. This implies that no target acquired before can be kept is the right part of the graph.

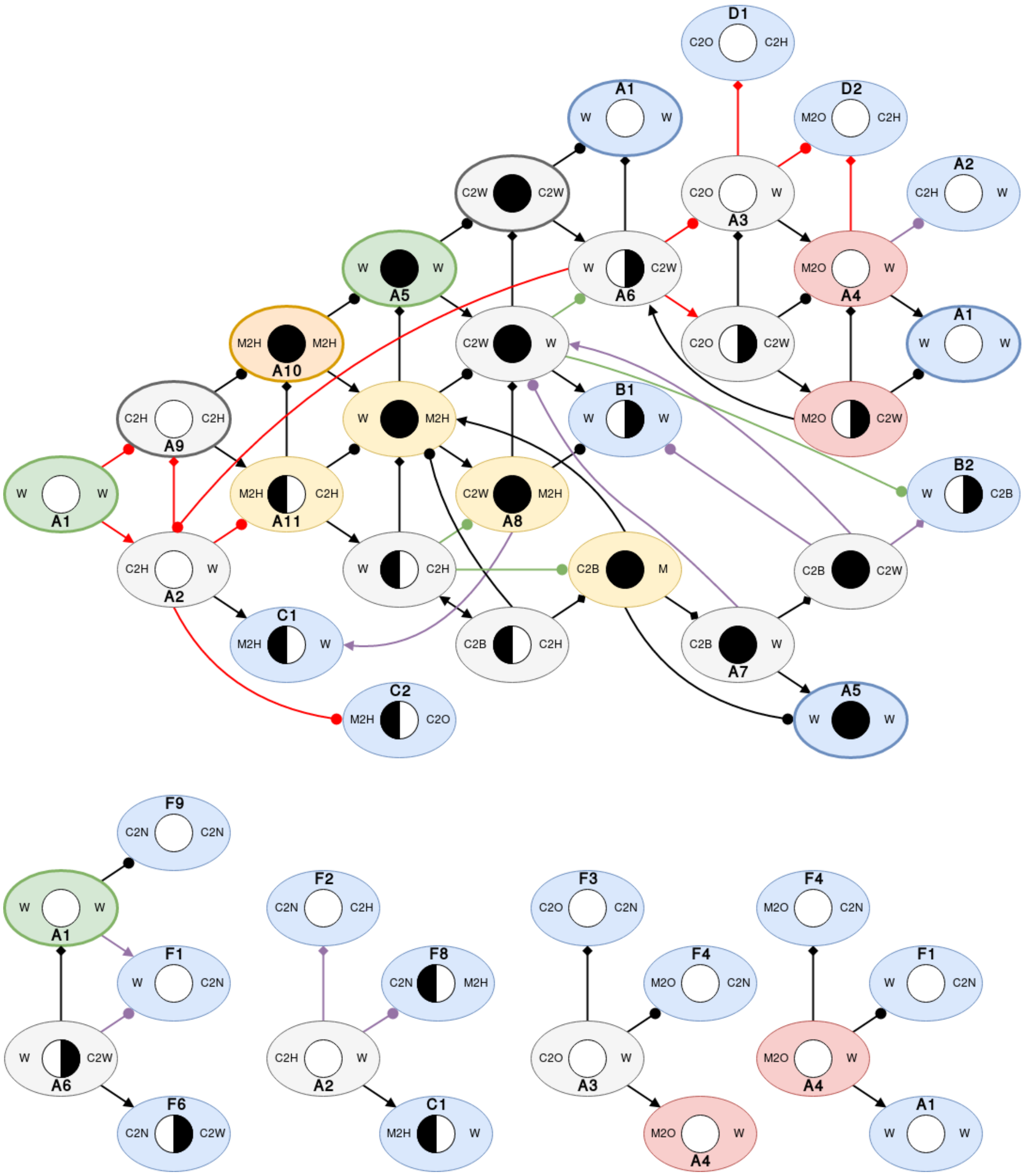

Figure 3: SYM configurations

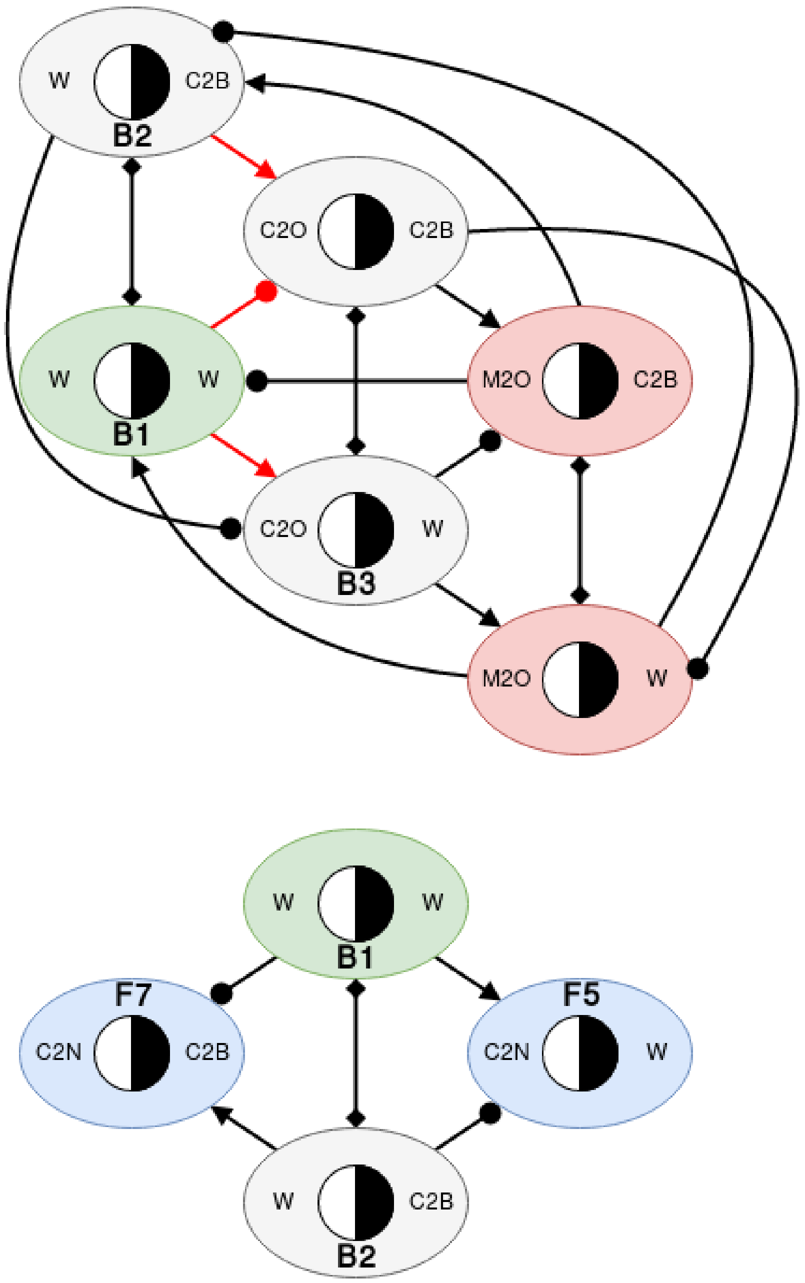

Figure 4: ASYM configurations

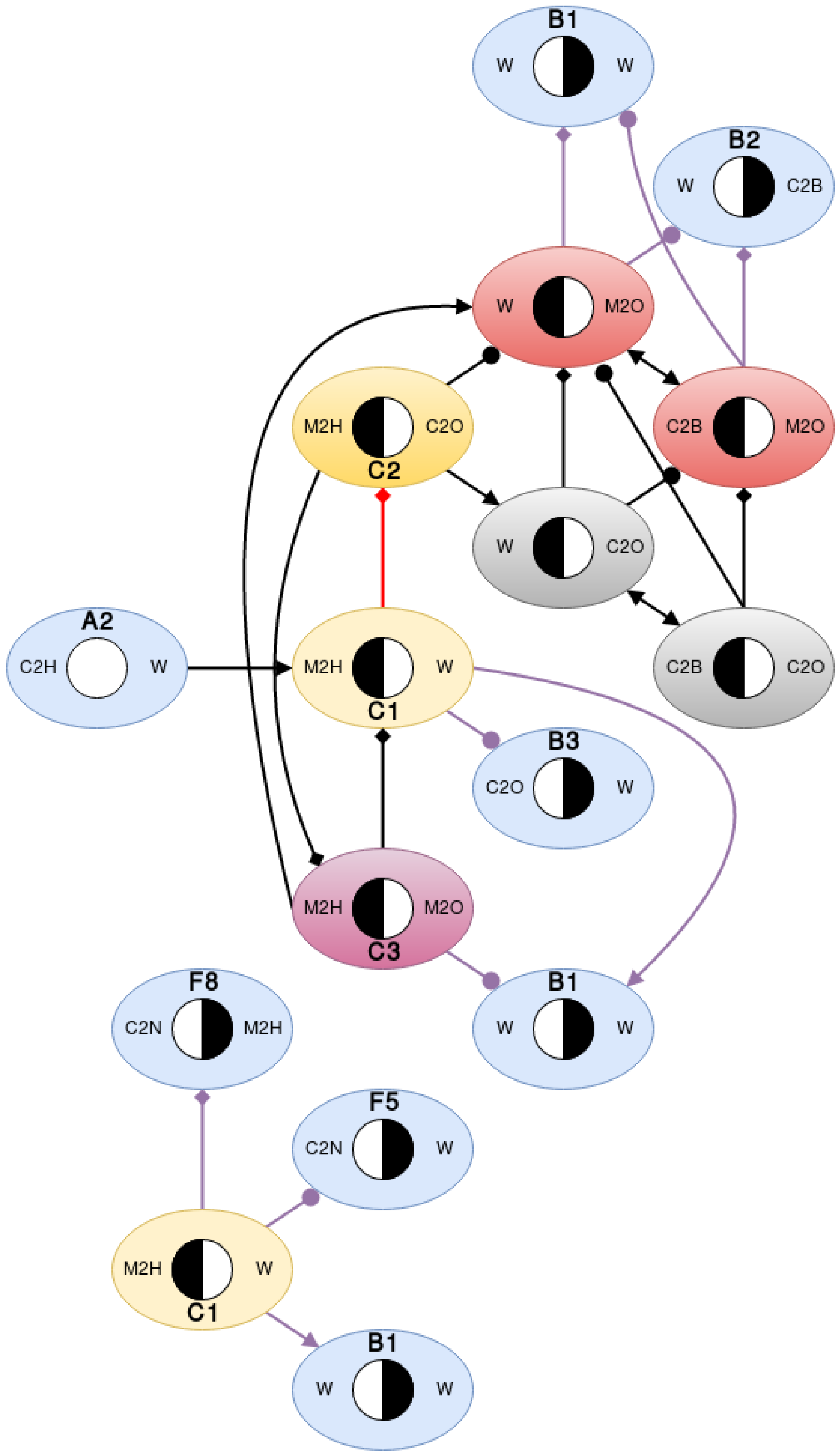

Figure 5: FAULTY 1 configurations

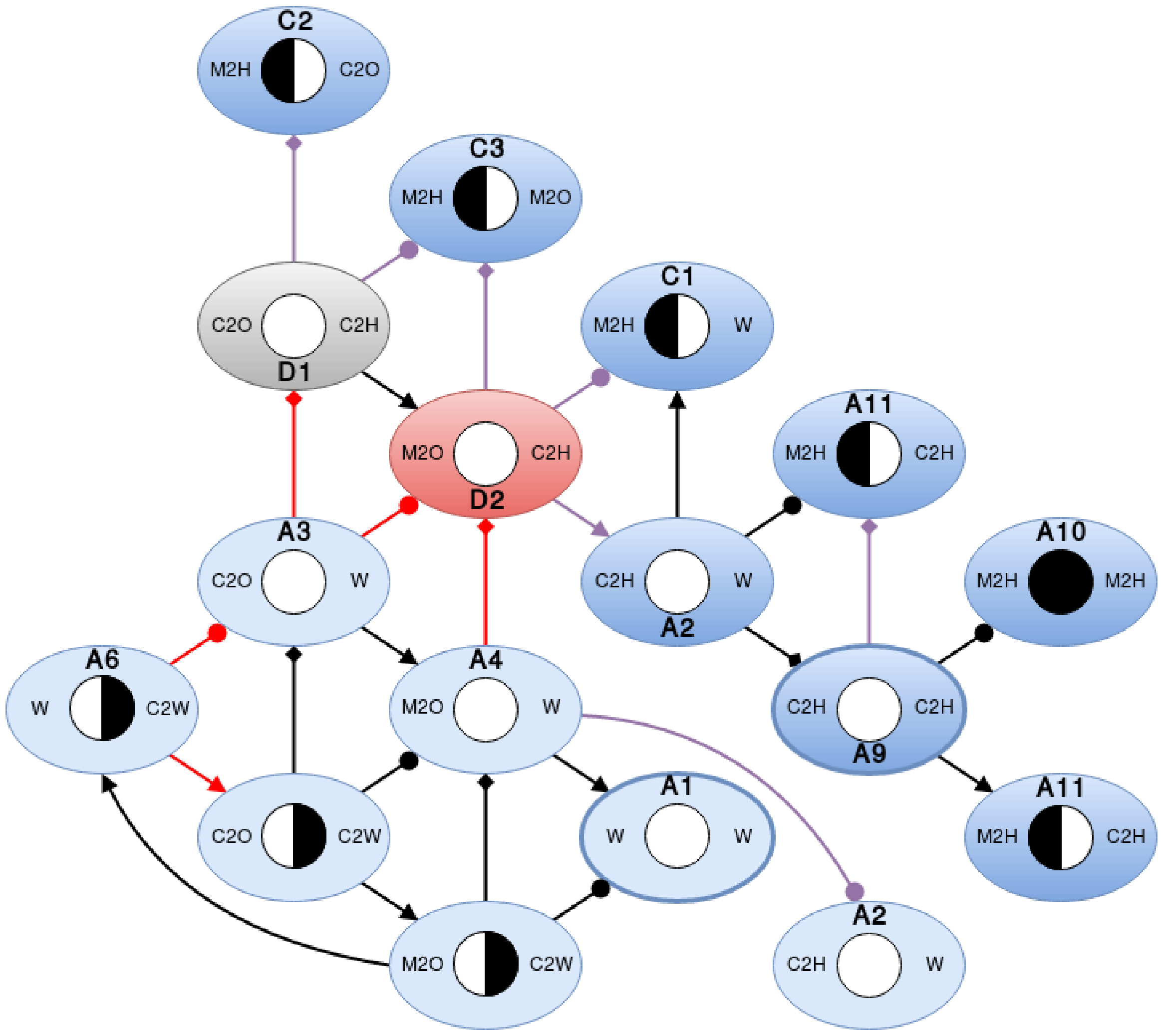

Figure 6: FAULTY 2 configurations

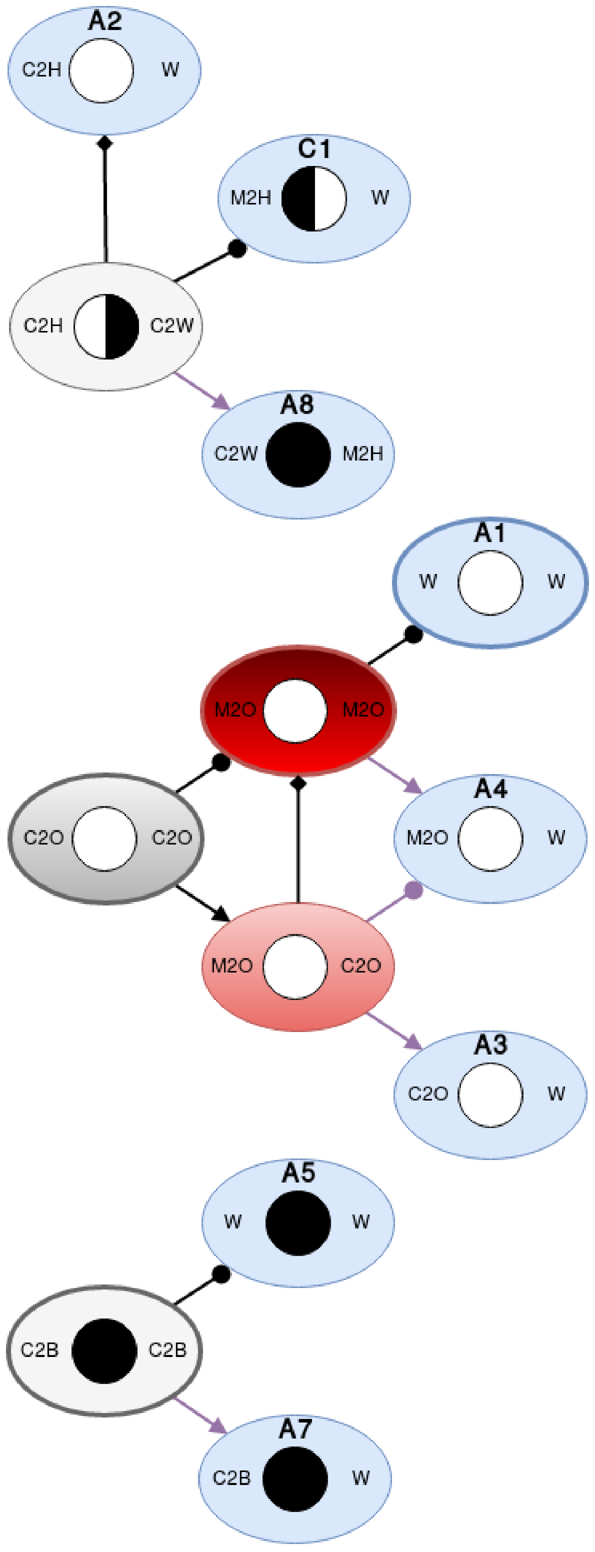

Figure 7: Illegal configurations

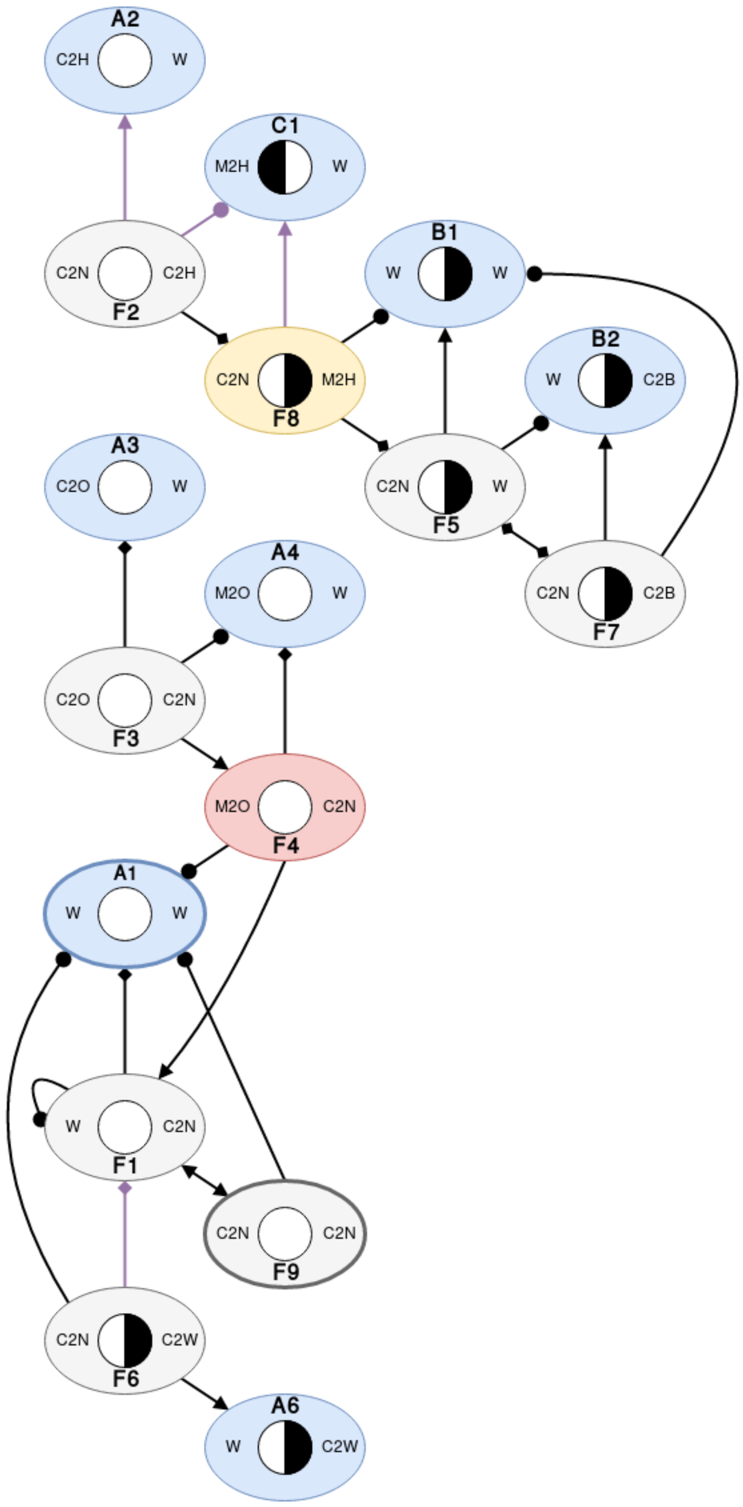

Figure 8: Gathered configurations

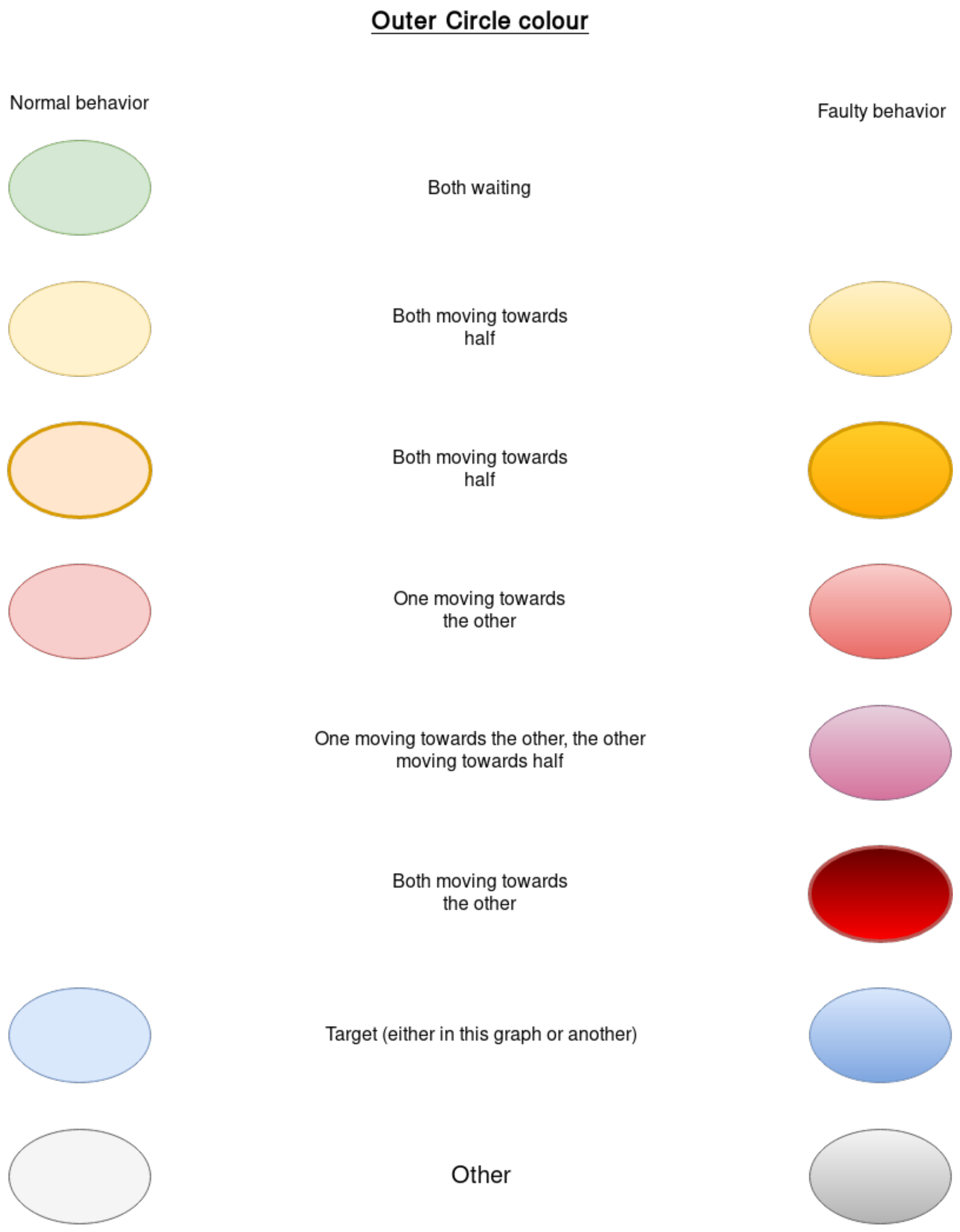
Outer Circle colour
Normal behavior
Faulty behavior
Both waiting
Both moving towards half
Both moving towards half
One moving towards the other
One moving towards the other, the other moving towards half
Both moving towards the other
Target (either in this graph or another)
Other

Figure 9: Graph legend

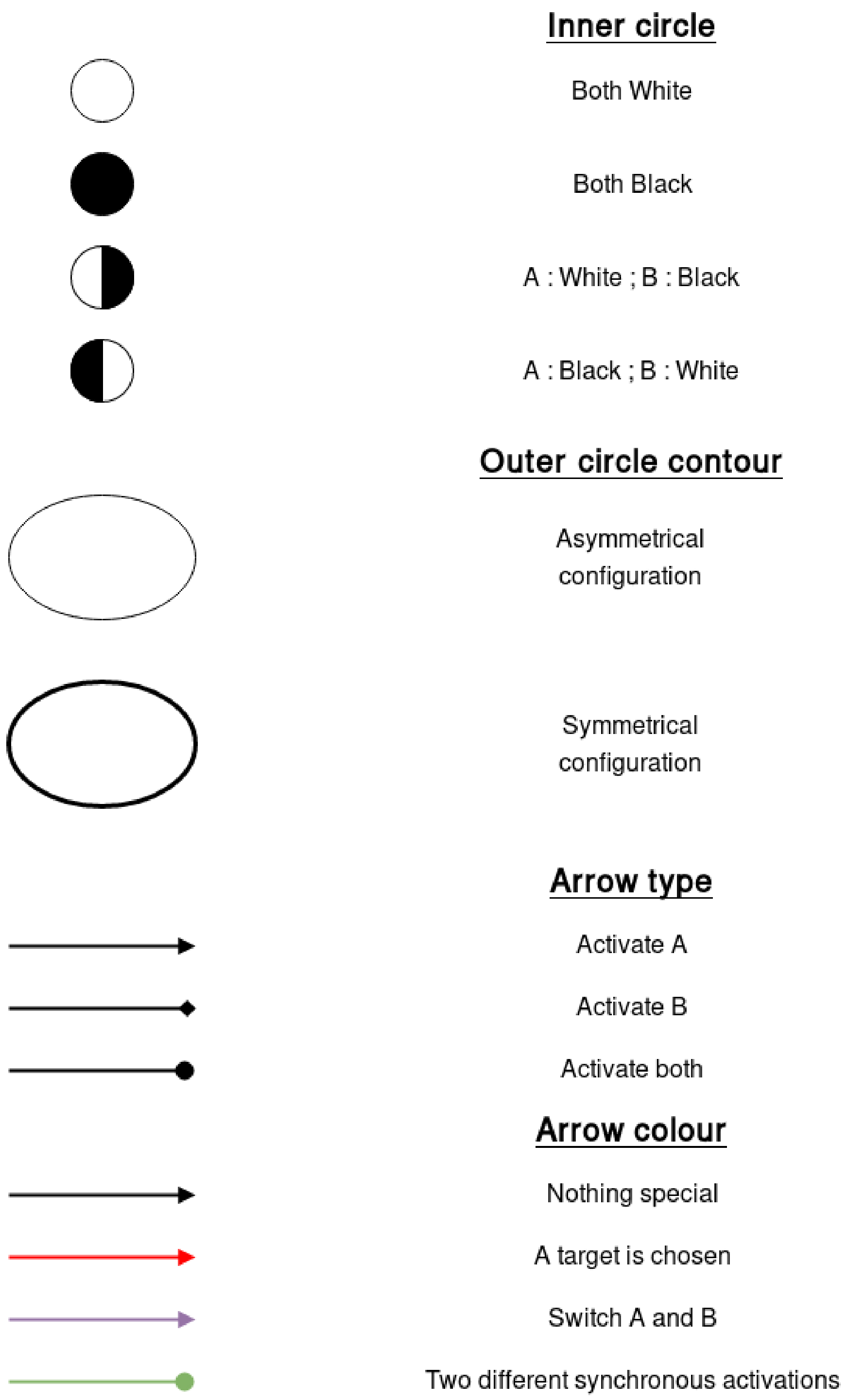

Figure 10: Graph Legend