Tokyo Tech

https://www.coord.c.titech.ac.jp

# Verification and Synthesis of Rendezvous Algorithms for Luminous Robots

*Moving and Computing (MAC) ・ Pisa, Tuscany, Italy ・ September 2022*

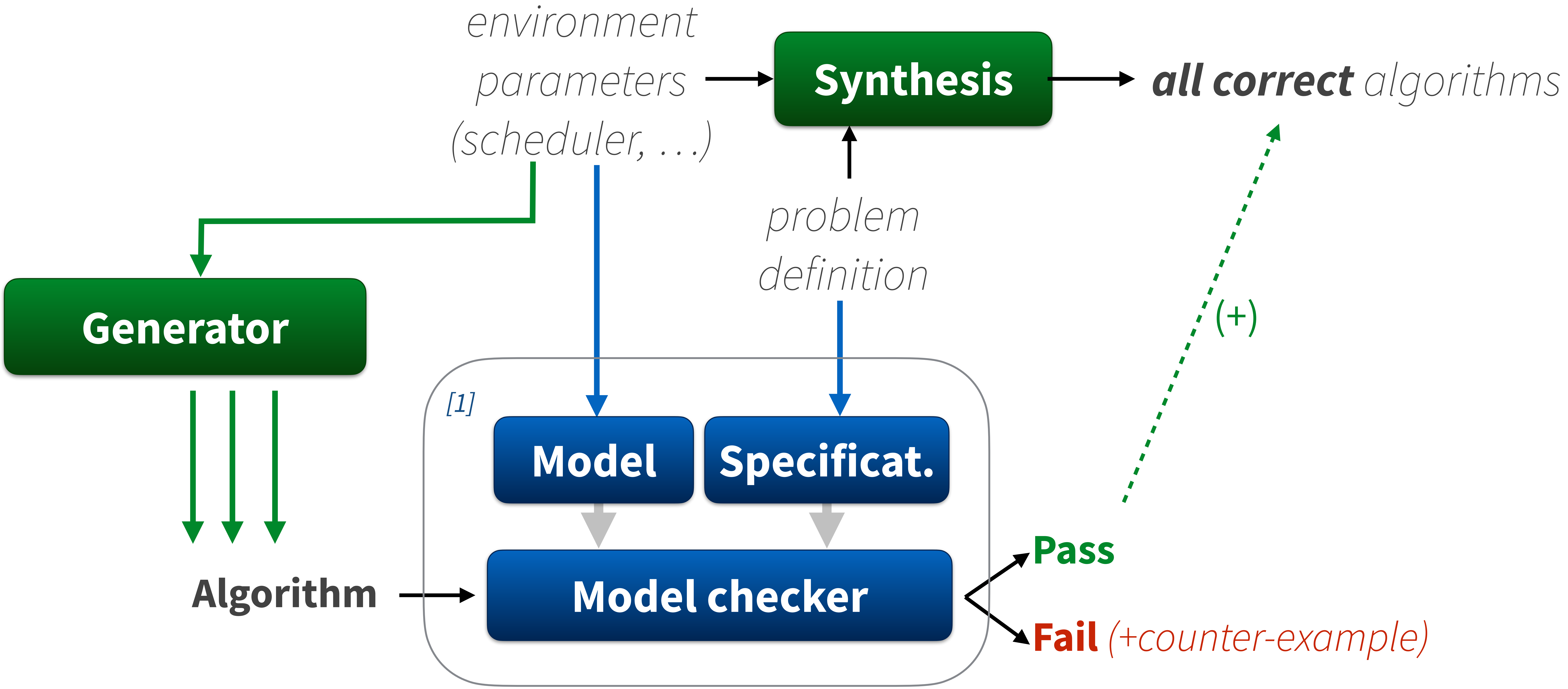

with

Sébastien Tixeuil and Koichi Wada

**Xavier Défago**

*Professor*
*School of Computing*
*Tokyo Institute of Technology*
*Japan*

# Overview




[1] D., Heriban, Tixeuil, Wada: **Using Model Checking to Formally Verify Rendezvous Algorithms for Robots with Lights in Euclidean Space**. SRDS 2020: 113-122

# Overview

- **Context**
  - 2 robots; oblivious, lights
- **Rendezvous**
  - arbitrary initial configuration
  - reach **same point**
  - … in **finite steps**
- **Related**
  - gathering problem

[1] Flocchini, Prencipe, Santoro: **Distributed Computing by Mobile Entities**. LNCS 11340 (2019)
[2] Giovanni Viglietta: **Rendezvous of Two Robots with Visible Bits**. ALGOSENSORS 2013: 291-306
[3] Suzuki, Yamashita: **Distributed anonymous mobile robots: Formation of geometric patterns**. SIAM J. Comp. 28(4): 1347-1363 (1999)
[4] D., Potop-Butucaru, Raipin-Parvédy. **Self-stabilizing gathering of mobile robots under crash and Byzantine faults**. Distrib. Comput. 2019.

# Model

[1] Suzuki, Yamashita: ***Distributed Anonymous Mobile Robots: Formation of Geometric Patterns***. *SIAM J. Comput. 28(4): 1347-1363 (1999)*
[2] Flocchini, Prencipe, Santoro: ***Distributed Computing by Mobile Entities***. *LNCS 11340 (2019)*

## Environment

- Euclidean (continuous)
- no common {origin, direction, unit distance, chirality}

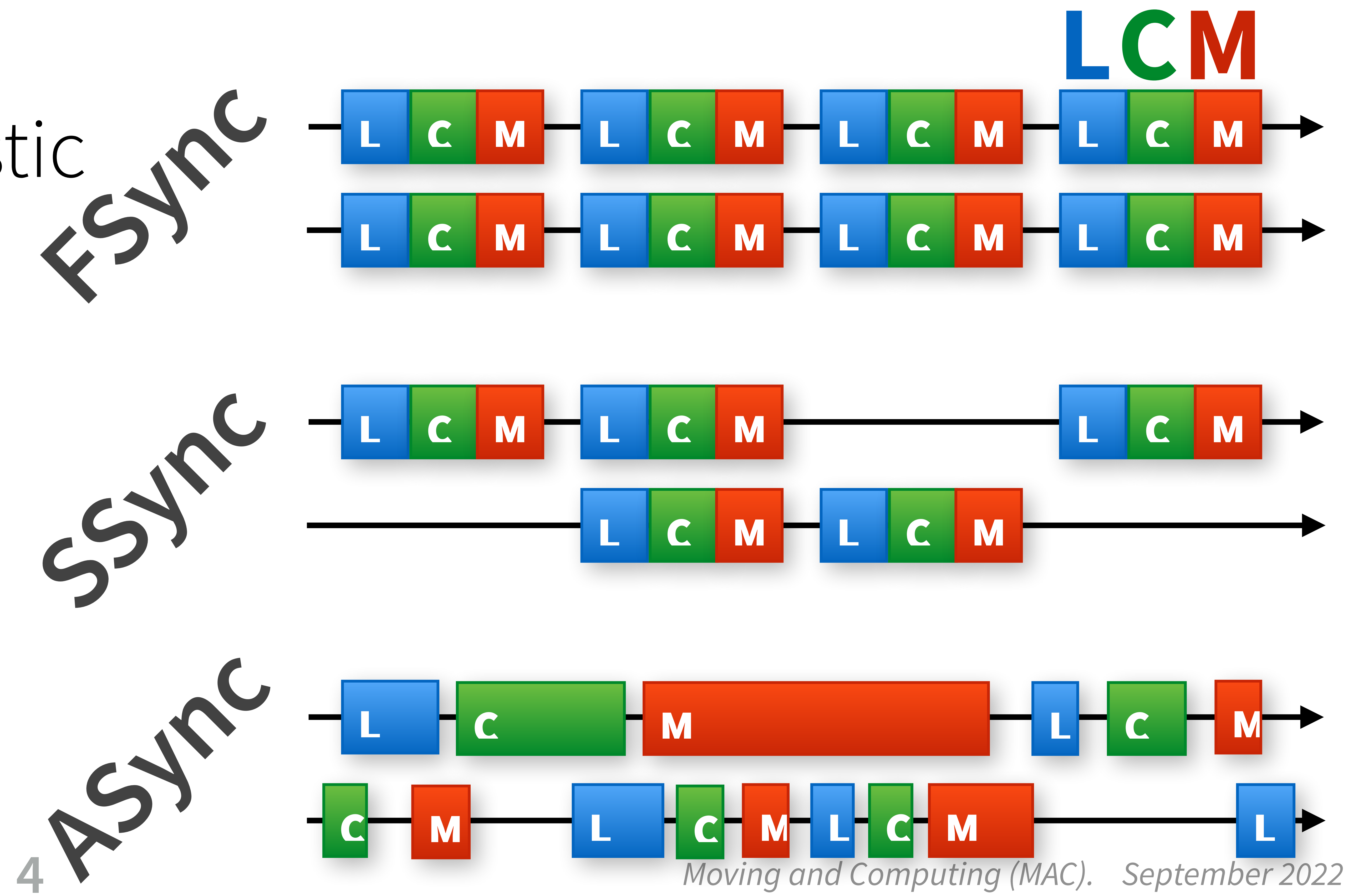
## Robots (OBLOT)

- oblivious, anonymous, deterministic

## Scheduler

- Look - Compute - Move
- synchrony models:

| | |
|---|---|
| FSync | fully synchronized |
| SSync | semi-synchronous |
| *ASync* | asynchronous |



# Luminous Robots

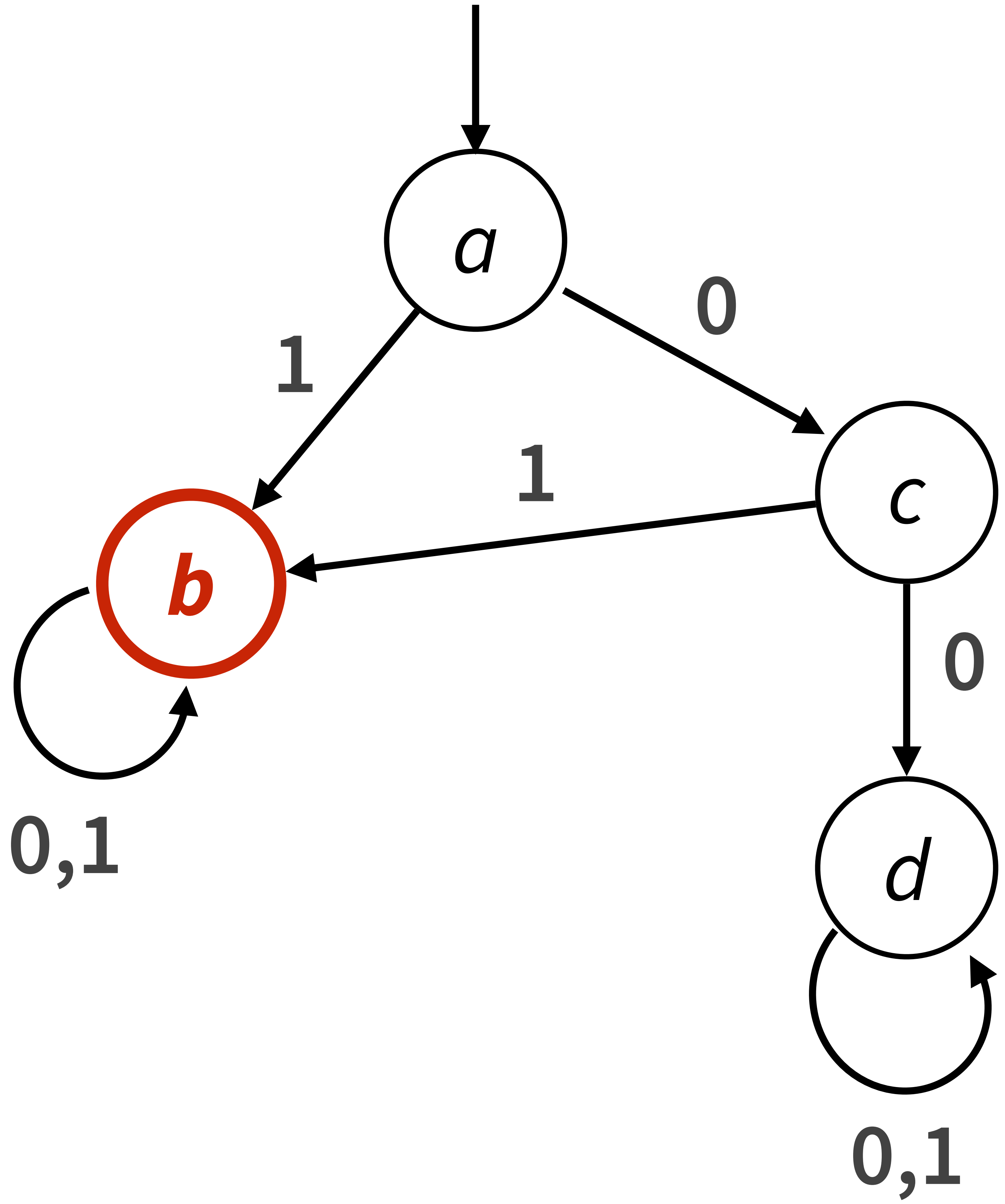
- **Full lights**
  - observes **own** light
  - observes **other's** light
- **External**
  - observe **other's** light only
- **Internal**
  - observe **own** light only

- **Class $\mathscr{L}$ [1]**
  - robots' colors
- **Non-$\mathscr{L}$**
  - robots' colors
  - relative position

*[1] Giovanni Viglietta: **Rendezvous of Two Robots with Visible Bits**. ALGOSENSORS 2013: 291-306*

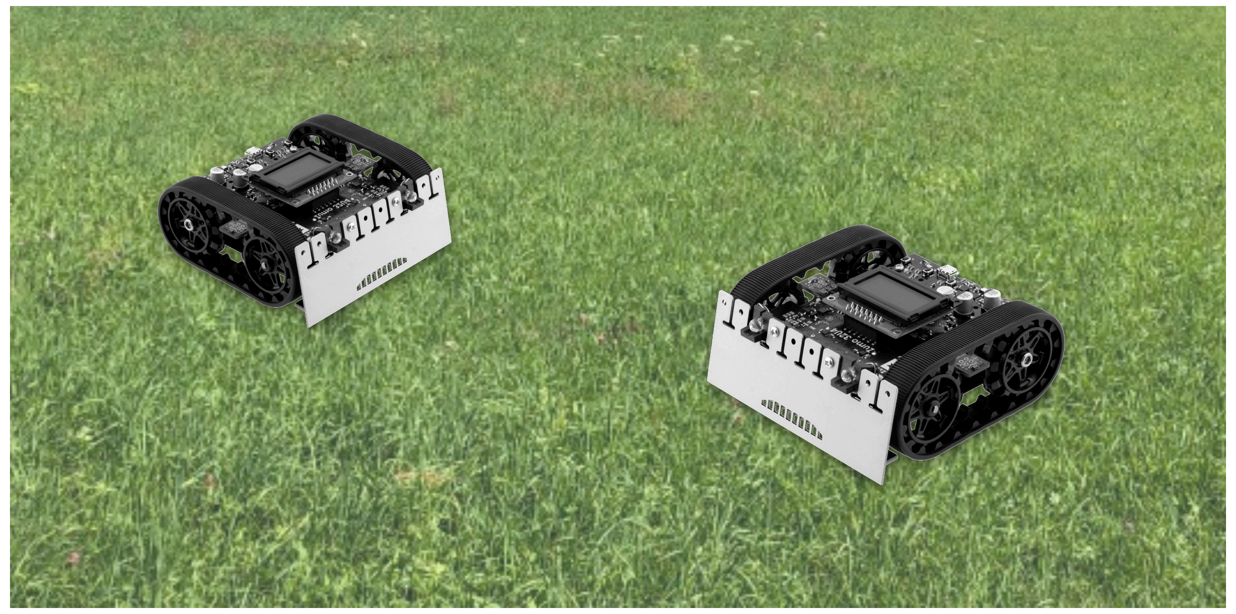
# Model Checking

- **Condition**
  - always not b
- **Model checker**
  - a0c0d0d... cycle on d **OK**
  - 1d... cycle on d **OK**
  - c1**b** reached b **STOP**
- **Counter-Example**
  - string: 01



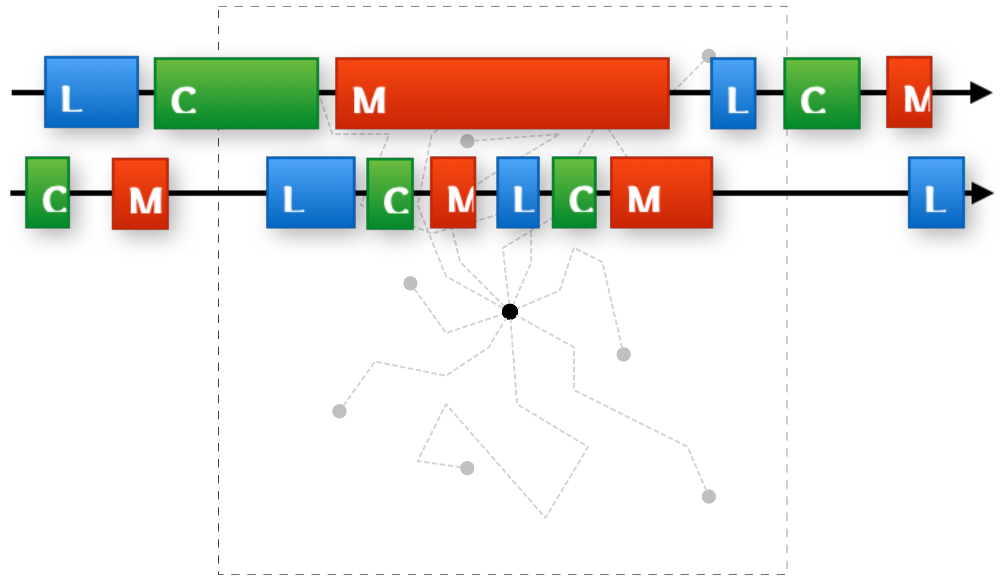
# Model Checking

## real-world



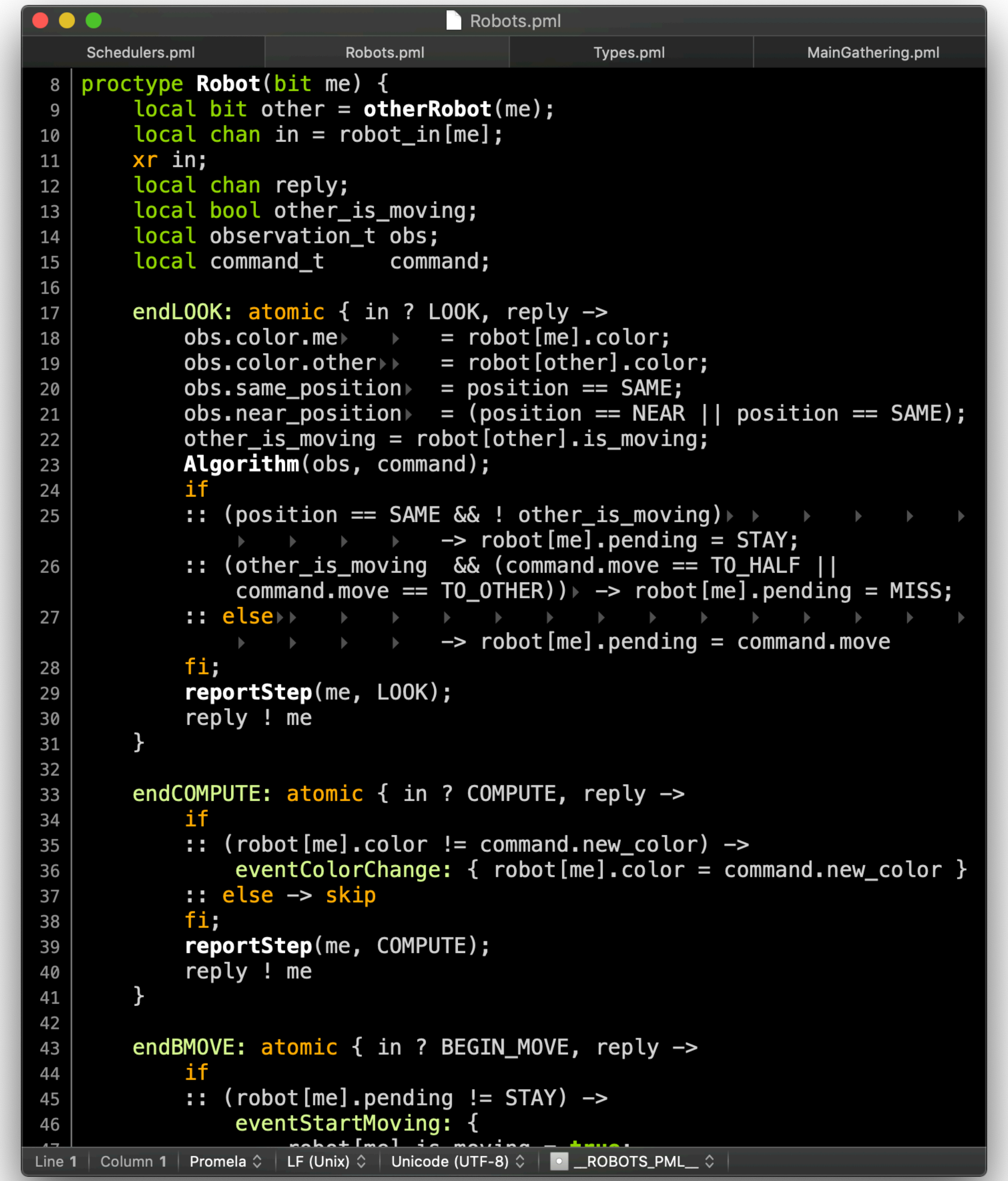
real robots
real environment
law of physics
Brownian motion

## robot model




robots = points
Euclidean space
mathematics
Zeno's paradox

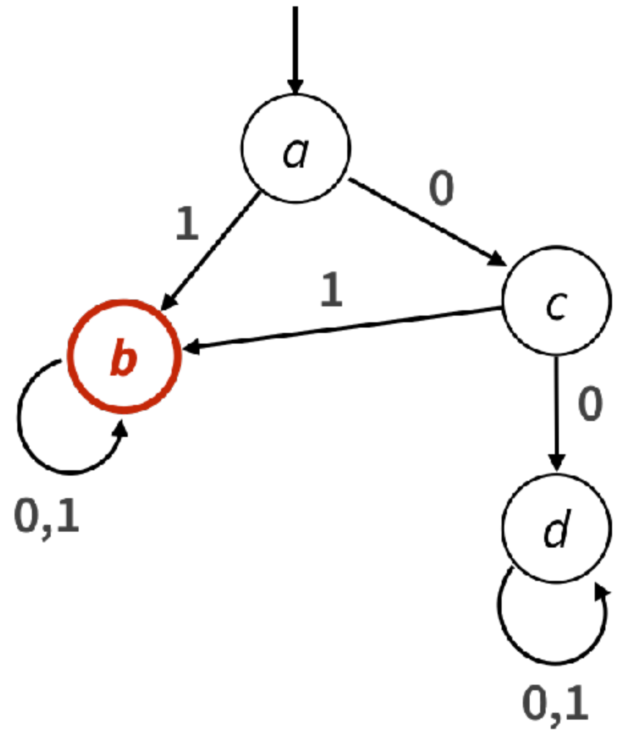
## verification model




```
proctype Robot(bit me) {
    local bit other = otherRobot(me);
    local chan in = robot_in[me];
    xr in;
    local chan reply;
    local bool other_is_moving;
    local observation_t obs;
    local command_t     command;

    endLOOK: atomic { in ? LOOK, reply ->
        obs.color.me        = robot[me].color;
        obs.color.other     = robot[other].color;
        obs.same_position   = position == SAME;
        obs.near_position   = (position == NEAR || position == SAME);
        other_is_moving = robot[other].is_moving;
        Algorithm(obs, command);
        if
        :: (position == SAME && ! other_is_moving)
                        -> robot[me].pending = STAY;
        :: (other_is_moving  && (command.move == TO_HALF ||
            command.move == TO_OTHER)) -> robot[me].pending = MISS;
        :: else
                        -> robot[me].pending = command.move
        fi;
        reportStep(me, LOOK);
        reply ! me
    }

    endCOMPUTE: atomic { in ? COMPUTE, reply ->
        if
        :: (robot[me].color != command.new_color) ->
            eventColorChange: { robot[me].color = command.new_color }
        :: else -> skip
        fi;
        reportStep(me, COMPUTE);
        reply ! me
    }

    endBMOVE: atomic { in ? BEGIN_MOVE, reply ->
        if
        :: (robot[me].pending != STAY) ->
            eventStartMoving: {
```

all dimensions finite
expressed as Promela code

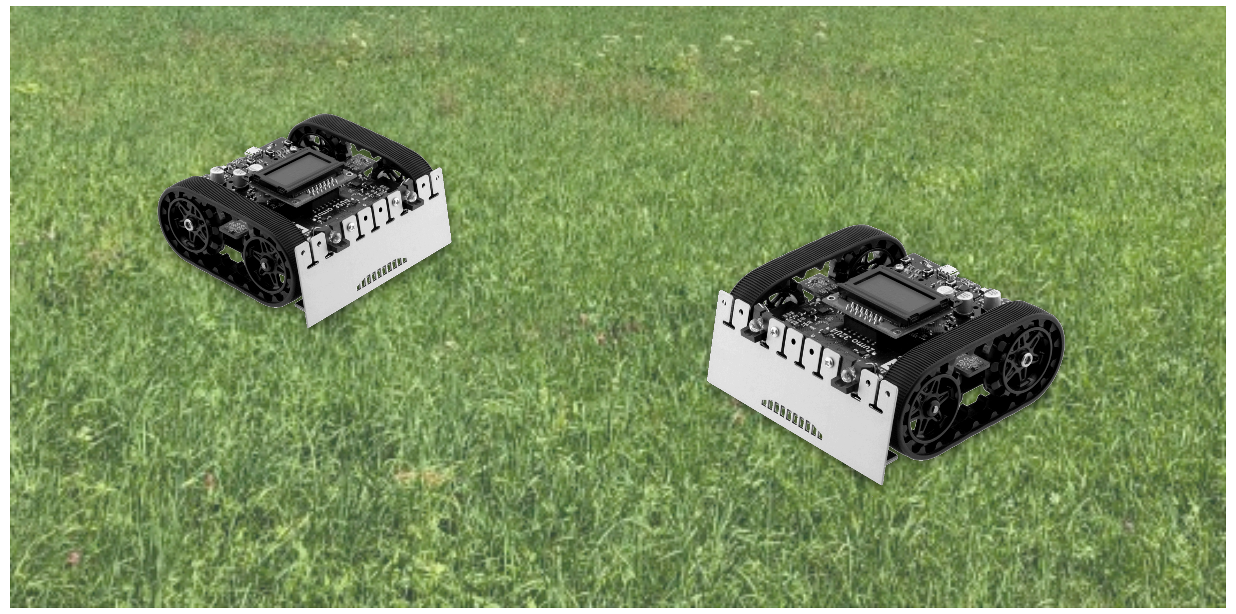
## model checker



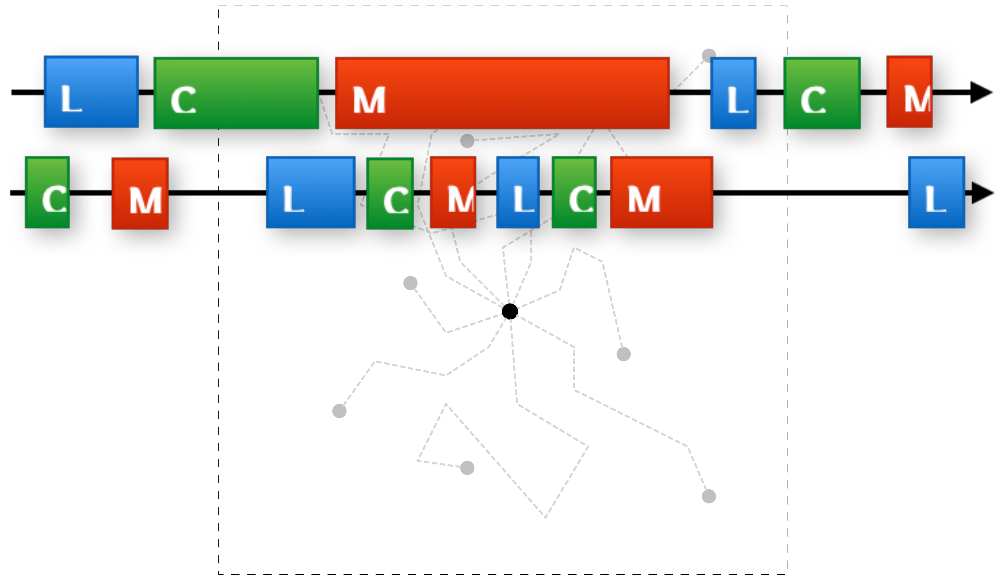

**loss of generality** → **loss of generality** →

# Model Checking

real-world

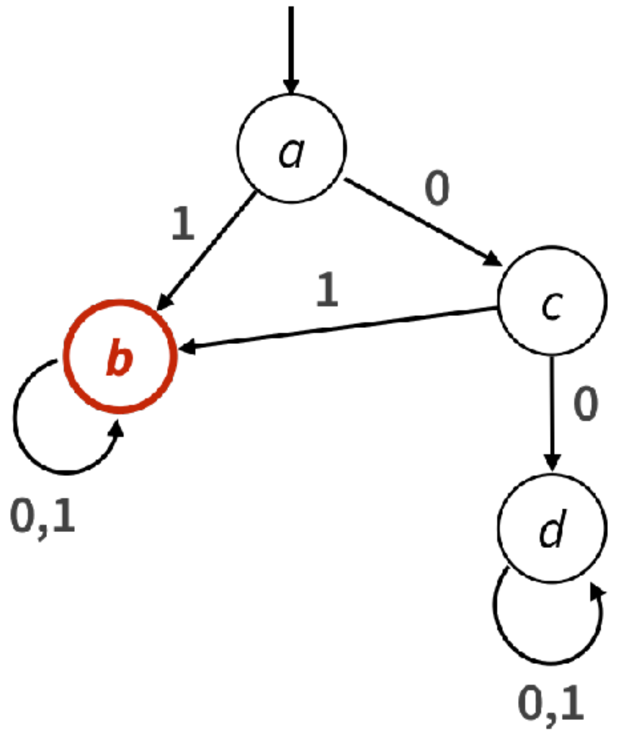


robot model




verification model


model checker




loss of generality

**loss of generality**

| Algorithm | | Verification |
|---|---|---|
| **faulty** | *exact* → | **FAIL !** |
| **correct** | *best-effort conservative* | **PASS** |

https://github.com/xdefago/spin-light

# Model-Checking

## `ltl gathering { <>[](position == SAME) }`

| Algorithm | ref. | Central. | FSync | SSync | LC-Atom | Mv-Atom | ASync |
|---|---|---|---|---|---|---|---|
| **NoMove** | triv. | *fail* | *fail* | *fail* | *fail* | *fail* | *fail* |
| **ToOther** | triv. | **pass** | *fail* | *fail* | *fail* | *fail* | *fail* |
| **ToHalf** | triv. | *fail* | **pass** | *fail* | *fail* | *fail* | *fail* |
| **Vig 2 cols** | [2] | **pass** | **pass** | **pass** | **pass** | *fail* | *fail* |
| **Vig 3 cols** | [2] | **pass** | **pass** | **pass** | **pass** | **pass** | **pass** |
| **Her 2 cols** | [3] | **pass** | **pass** | **pass** | **pass** | **pass** | **pass** |
| **Flo 3 cols X** | [1] | **pass** | **pass** | **pass** | *fail* | *fail* | *fail* |
| **Oku 5 cols X** | [5] | **pass** | **pass** | **pass** | **pass** | *fail* | *fail* |
| **Oku 4 cols X** | [5] | **pass** | *fail* | *fail* | *fail* | *fail* | *fail* |
| **Oku 3 cols X** | [5] | **pass** | *fail* | *fail* | *fail* | *fail* | *fail* |


*[1] P. Flocchini, N. Santoro, G. Viglietta, and M. Yamashita. **Rendezvous with constant memory**. Theor. Comput. Sci., 621(C):57–72, March 2016.*

*[2] G. Viglietta. **Rendezvous of two robots with visible bits**. In Proc. 9th ALGOSENSORS, pp. 291–306, 2014.*

*[3] A. Heriban, X. Défago, S. Tixeuil. **Optimally gathering two robots**. In Proc. 19th ICDCN, Jan. 2018.*

*[4] T. Okumura, K. Wada, Y. Katayama. **Optimal asynchronous rendezvous for mobile robots with lights**. In Proc. 19th SSS, Nov. 2017.*

*[5] T. Okumura, K. Wada, X. Défago. **Optimal rendezvous L-algorithms for asynchronous mobile robots with external lights**. In Proc. 22nd OPODIS, Dec. 2018.*

# Algorithm Description

[ Full colors ]

**(col A) (col B) (same?) —> (move) (col)**

*own color* *other's color* *same location ?* *new color*

Stay,
toHalf,
toOther

## Encoding

**guards[]** **actions[]**

00s_01s_10s_11s_00d_01d_10d_11d__S1_S0_S1_S0_S1_S0_O1_H0

**01d** -> **S0**

(my = 0) (other = 1) (¬ gathered) —> Stay & new := 0

# Algorithm Description

[ Full colors class L ]

**(col A) (col B) ~~(same?)~~ —> (move) (col)**

*own color* *other's color* ~~*same location ?*~~ *new color*

Stay,
toHalf,
toOther

## Encoding

**guards[]** | **actions[]**

~~00s_01s_10s_11s_00d~~_01d_10d_11d__S1_S0_~~S1_S0_S1_S0_O1~~_H0

00_01_10_11__S1_S0_O1_H0

# Algorithm Description

[ External colors ]

~~(col A)~~ (col B) (same?) —> (move) (col)

*~~own color~~* *other's color* *same location ?* Stay, toHalf, toOther *new color*

## Encoding

guards[] | actions[]

00s_01s_10s_11s_00d_01d_10d_11d__S1_S0_S1_S0_S1_S0_O1_H0

0s_1s_0d_1d__S1_S0_S1_S0

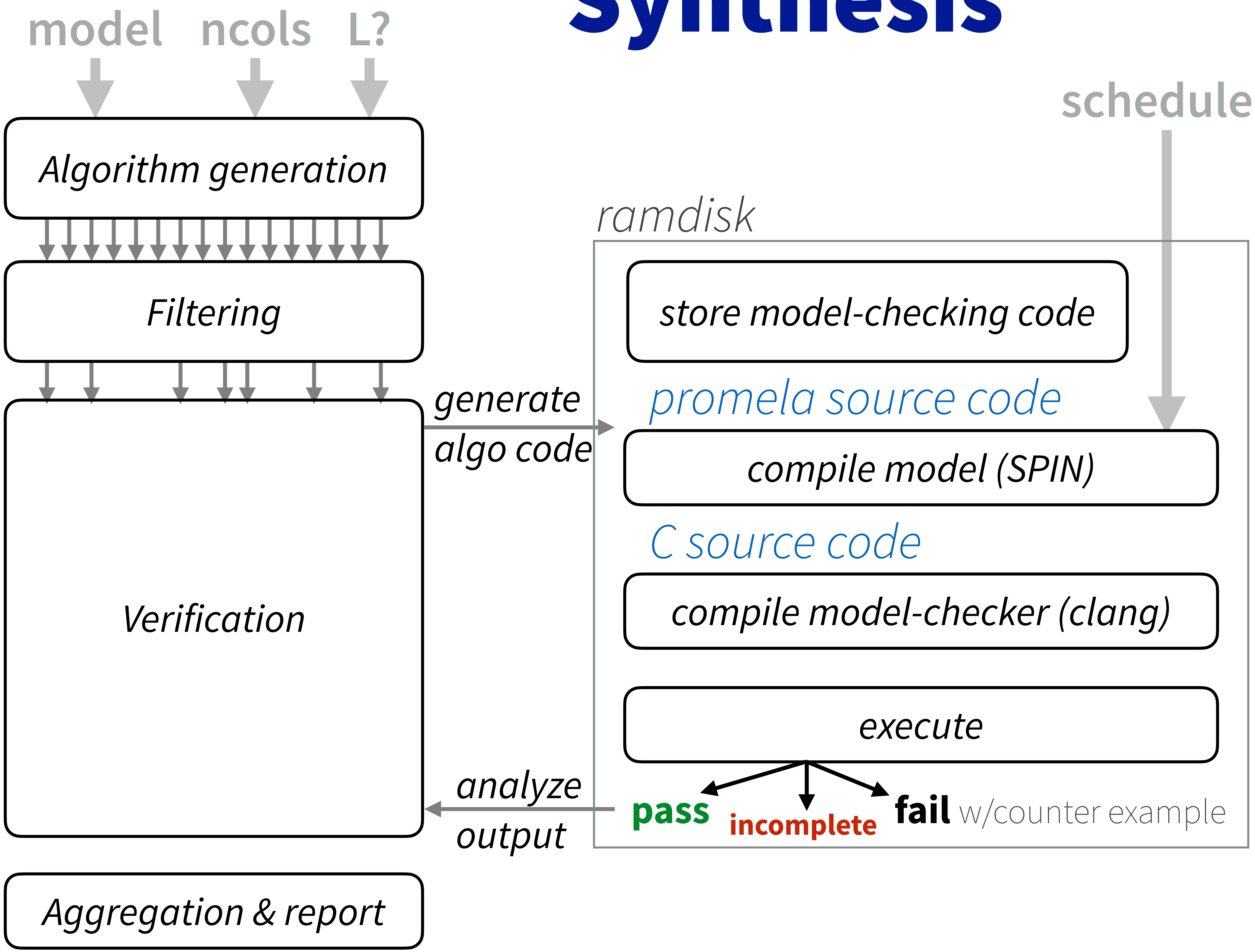
# Algorithm Description

00s_01s_10s_11s_00d_01d_10d_11d__S1_S0_S1_S0_S1_S0_O1_H0

## Generated algorithm *(promela code)*

```
# define ALGO_NAME     "ALGO_SYNTH_00s_01s_10s_11s_00d_01d_10d_11d__S1_S0_S1_S0_S1_S0_O1_H0"
# define Algorithm(o,c) Alg_Synth(o,c)
# define MAX_COLOR     (2)
# define NUM_COLORS    (2)
inline Alg_Synth(obs, command)
{
  command.move     = STAY;
  command.new_color = obs.color.me;
  if
  :: (obs.color.me == 0) && (obs.color.other == 0) && (obs.same_position) -> command.move = STAY; command.new_color = 1;
  :: (obs.color.me == 0) && (obs.color.other == 1) && (obs.same_position) -> command.move = STAY; command.new_color = 0;
  :: (obs.color.me == 1) && (obs.color.other == 0) && (obs.same_position) -> command.move = STAY; command.new_color = 1;
  :: (obs.color.me == 1) && (obs.color.other == 1) && (obs.same_position) -> command.move = STAY; command.new_color = 0;
  :: (obs.color.me == 0) && (obs.color.other == 0) && ! (obs.same_position) -> command.move = STAY; command.new_color = 1;
  :: (obs.color.me == 0) && (obs.color.other == 1) && ! (obs.same_position) -> command.move = STAY; command.new_color = 0;
  :: (obs.color.me == 1) && (obs.color.other == 0) && ! (obs.same_position) -> command.move = TO_OTHER; command.new_color = 1;
  :: (obs.color.me == 1) && (obs.color.other == 1) && ! (obs.same_position) -> command.move = TO_HALF; command.new_color = 0;
  fi;
}
```

# Synthesis


model
ncols
L?
Algorithm generation
Filtering
Verification
Aggregation & report
generate
algo code
analyze
output
scheduler
ramdisk
store model-checking code
promela source code
compile model (SPIN)
C source code
compile model-checker (clang)
execute
pass
incomplete
fail w/counter example

# Synthesis Execution

% synth-lights -s ssync full 2 -R

*SSYNC scheduler*

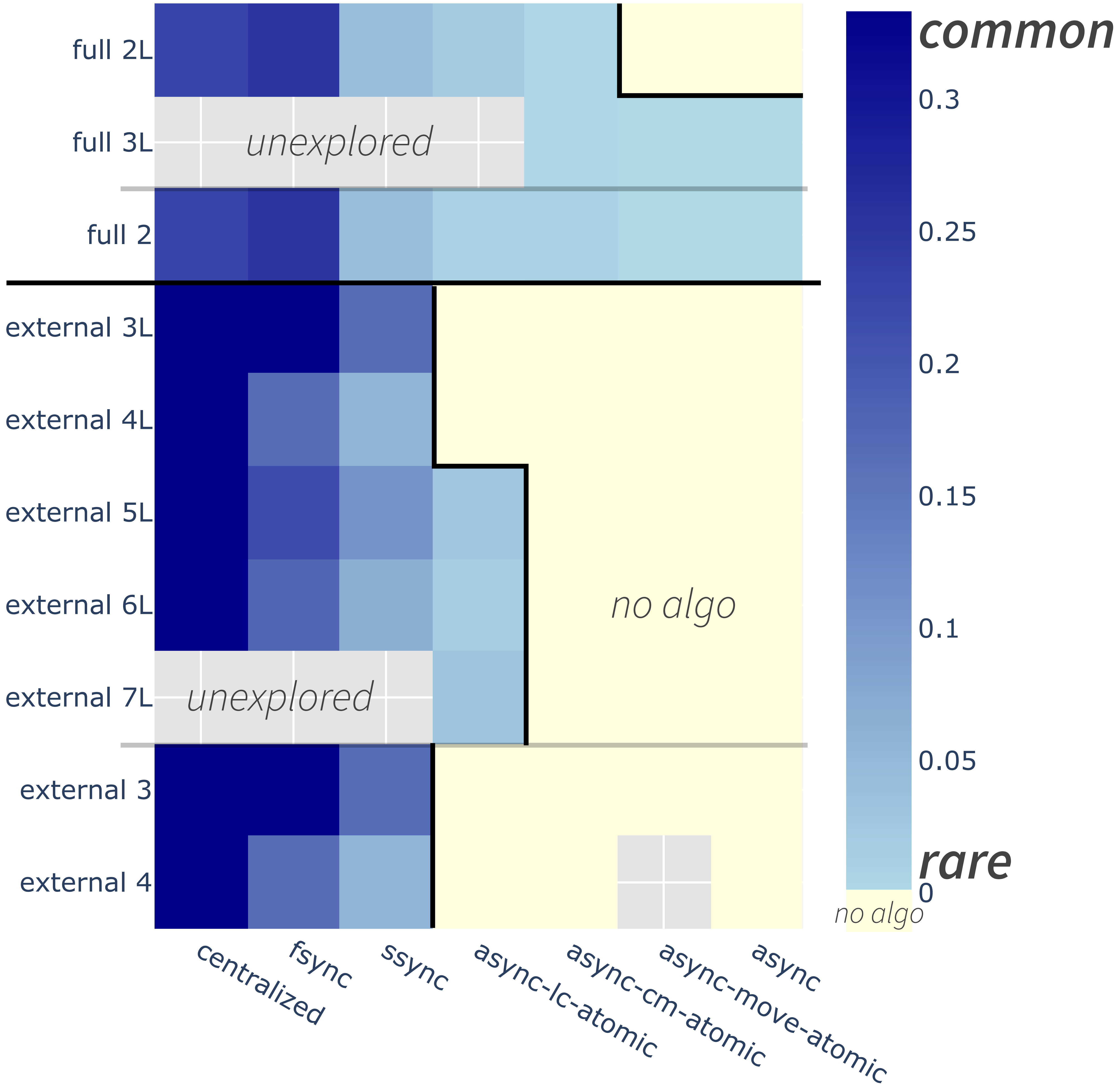
*full (2) colors*

*apply Viglietta's retain rule*

```
arm √ Rust/synth-lights % ~/.cargo/target/release/synth-lights -s ssync full 2 -R
```

# Synthesis (Outcomes)

| Centralized | FSYNC | SSYNC | LC-atomic ASYNC | CM-atomic ASYNC | Move-atomic ASYNC | ASYNC | |
|---|---|---|---|---|---|---|---|
| 66 | 74 | 12 | 6 | 1 | - | - | full 2 $\mathcal{L}$ |
| | | | | | | 4 | full 3 $\mathcal{L}$ |
| 1056 | 1184 | 192 | 36 | 44 | 2 | 2 | full 2 |
| 2 | 2 | 1 | - | - | - | - | external 3 $\mathcal{L}$ |
| 24 | 12 | 4 | - | - | - | - | external 4 $\mathcal{L}$ |
| 240 | 156 | 78 | 20 | - | - | - | external 5 $\mathcal{L}$ |
| 2400 | 1264 | 464 | 108 | - | - | - | external 6 $\mathcal{L}$ |
| | | | 2232 | - | | - | external 7 $\mathcal{L}$ |
| 54 | 54 | 27 | - | - | - | - | external 3 |
| 6144 | 3072 | 1024 | - | - | | - | external 4 |



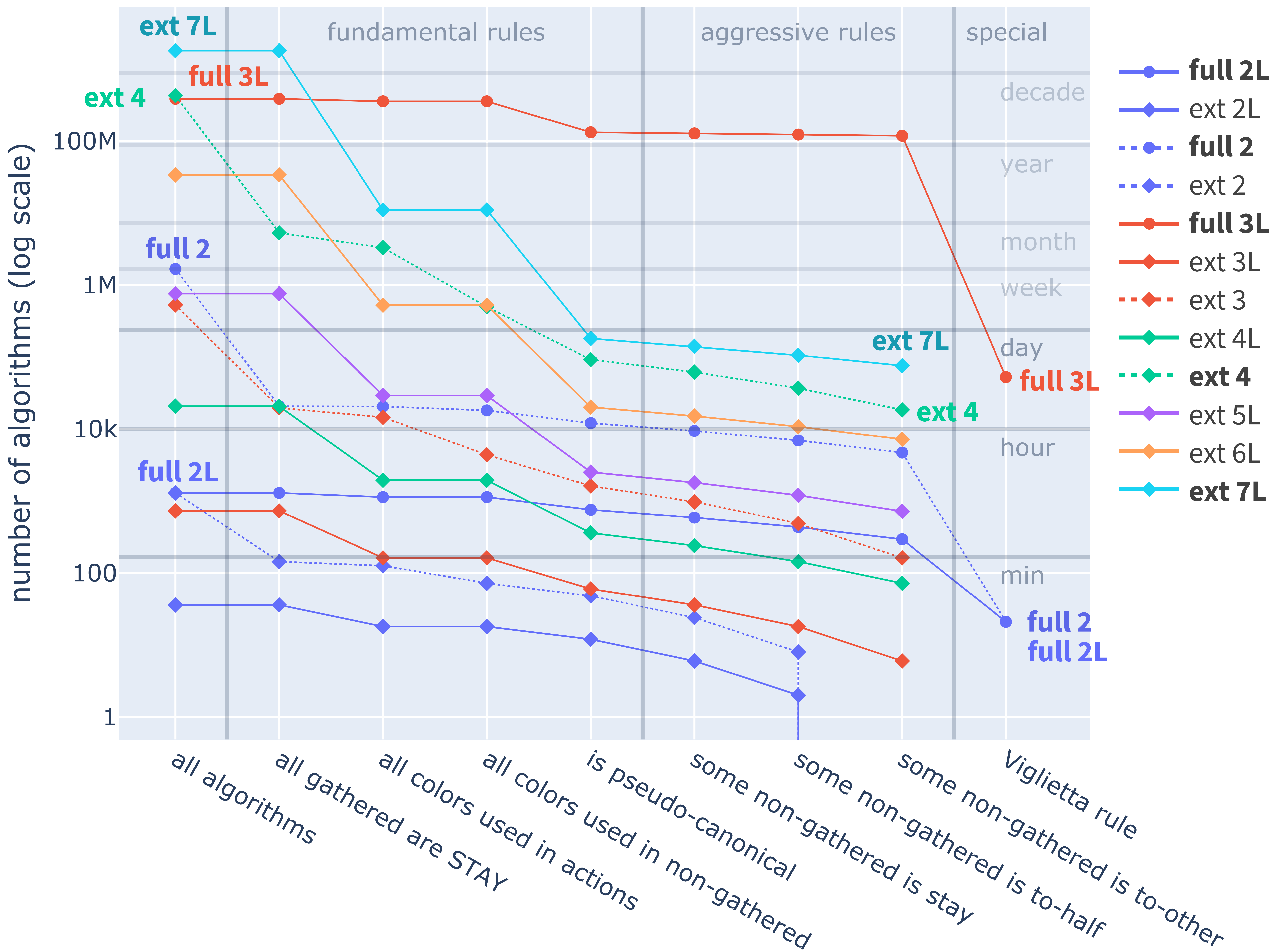
# Filtering Rules

Tokyo Tech

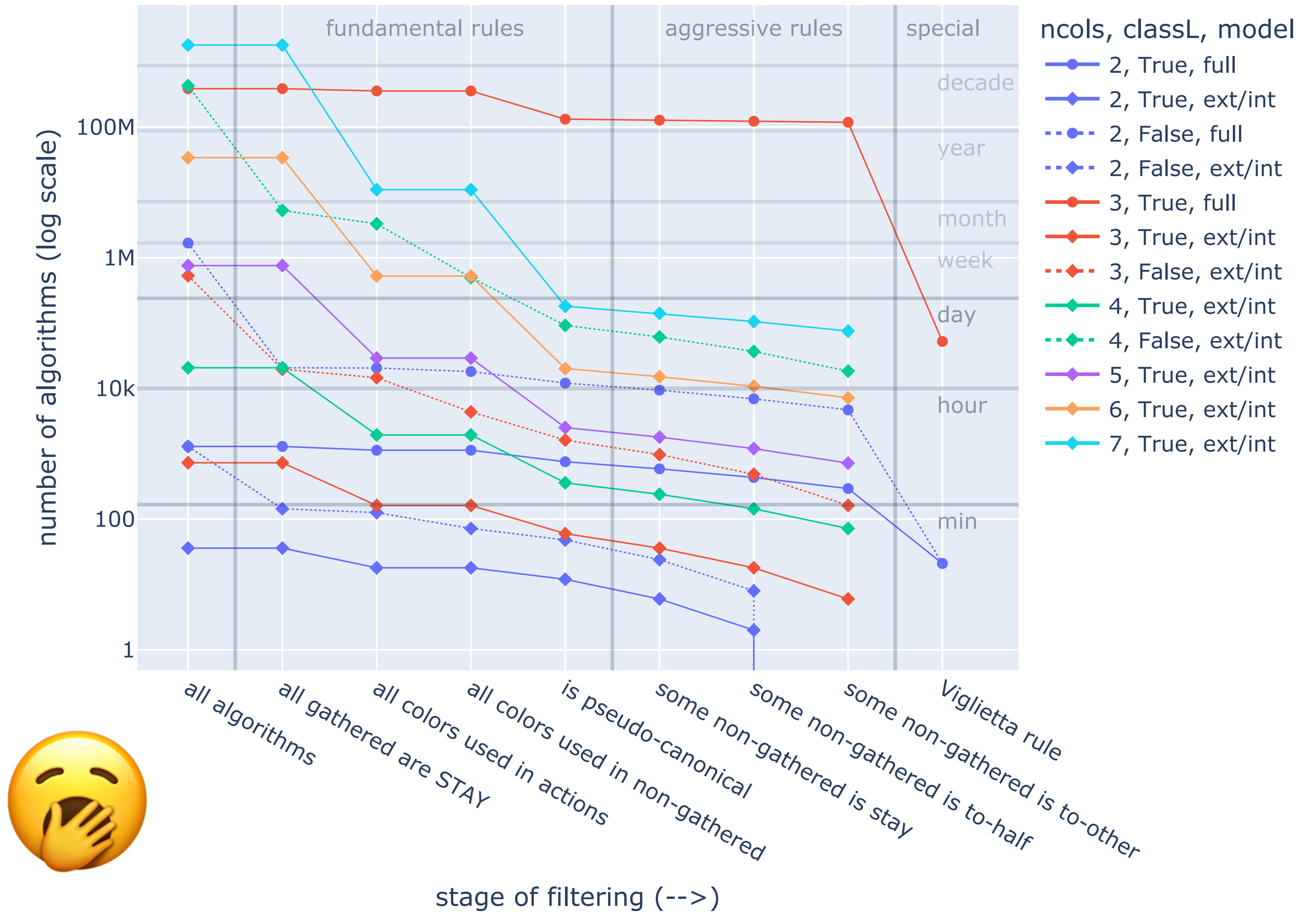
- **Fundamental**
  - gathered => STAY
  - all colors used in some action
  - all colors used in non-gathered action
  - pseudo-canonical (reduce symmetries)
- **Aggressive**
  - some non-gathered is STAY; toHALF; toOTHER
- **Viglietta (retain rule)**
  - *"A robot retains its color iif it sees the other robot set to a diff. color."*

[3] Giovanni Viglietta: **Rendezvous of Two Robots with Visible Bits**. ALGOSENSORS 2013: 291-306

# Filtering Rules



# Filtering Rules




## The Frontier... (algo. enumeration)

- enumeration only
  ~ 400,000 algos / s
- as reference
  external/internal 7L 1,801,088,541 (~1.5 h) 🥱
- less tractable
  external/internal 8L 110,075,314,176 (~ 3 days) 😴
  external/internal 5 576,650,390,625 (~ 1/2 month) 😵
  full 3 150,094,635,296,999,121 (~ 12,000 years) 💀
  full 4L 184,884,258,895,036,416 (~ 14,650 years)

# Conclusion

## Outcomes

- feasible up to:
  - full **2**, full **3L**,
  - int/ext **4**, int/ext **7L**
- consistent with known results
- allows quick exploration

## Method

- conservative (pessimistic)
- quite accurate

## Future

- optimize generation
- better filtering
- model variants (quasiSS,…)
- optimistic model
- adapt to n>2 robots

## Software

- github projects
- public after conference