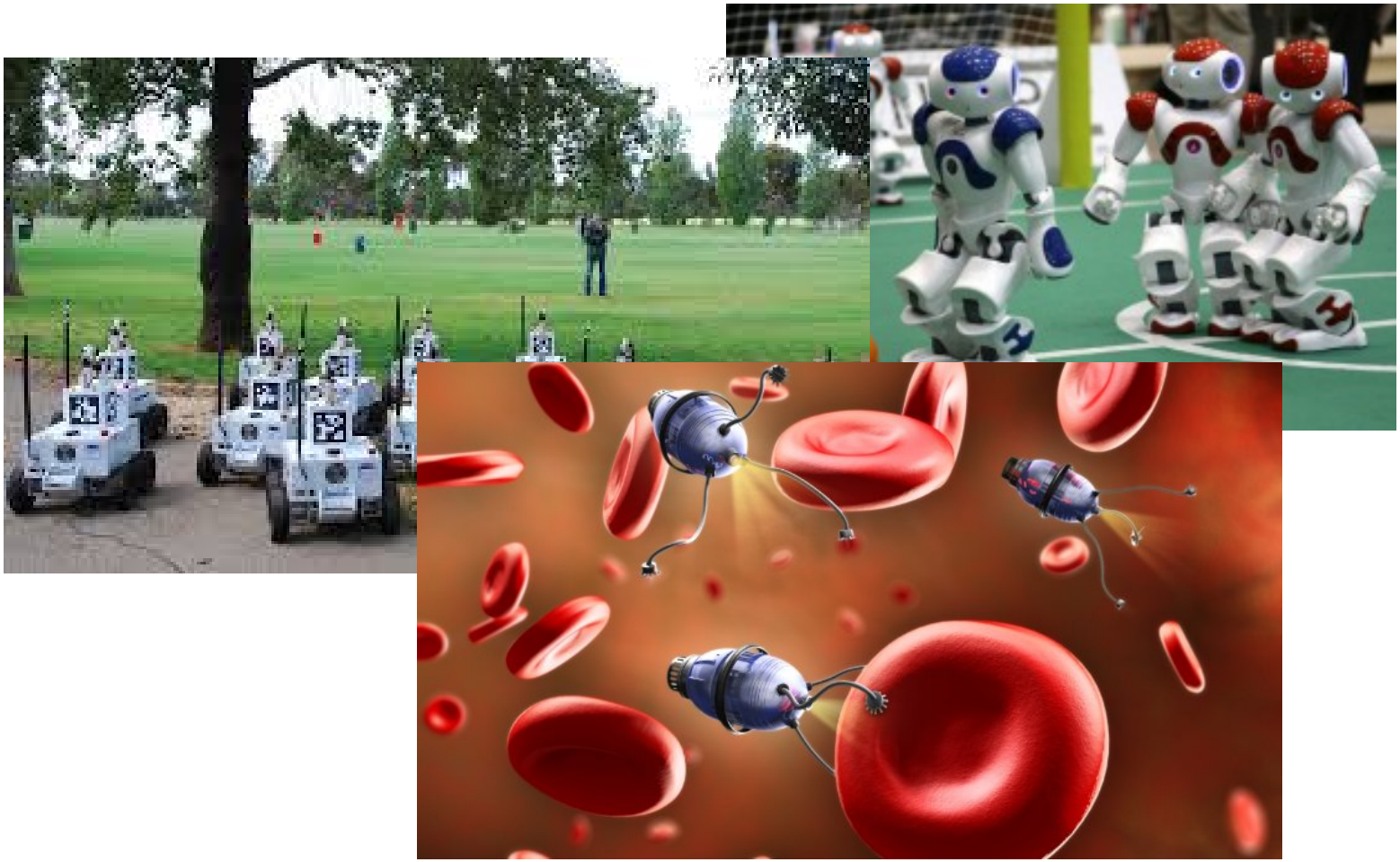
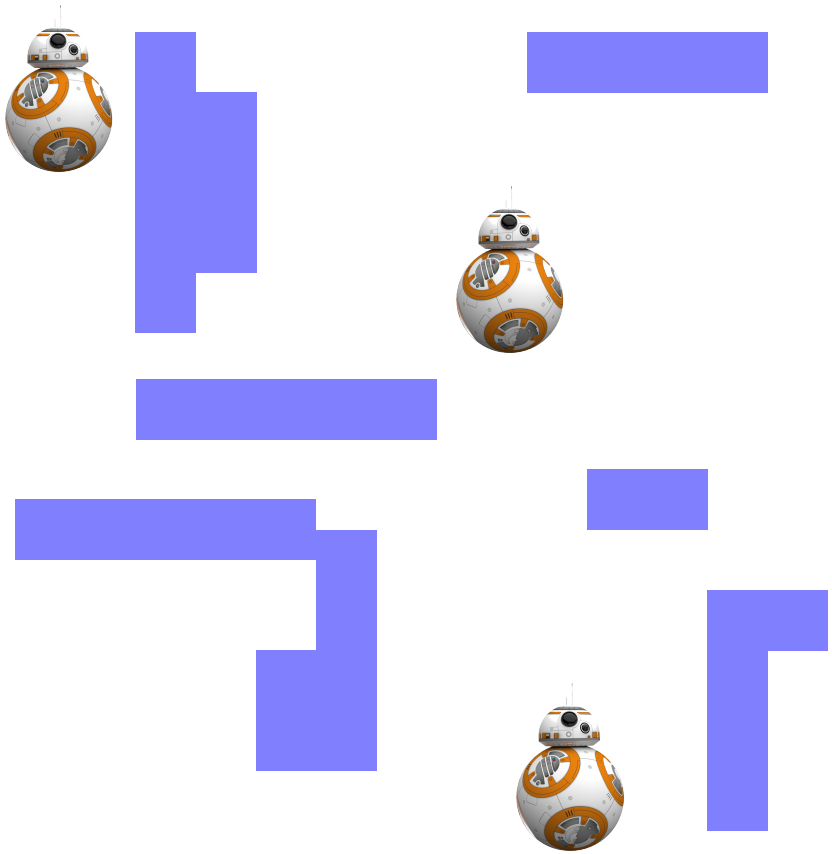
Verification and Synthesis of robot protocols

Laure Millet
advised by: Béatrice Bérard and Maria Potop-Butucaru

Robot teams: a mobile distributed system



Dumb robots [SY93]

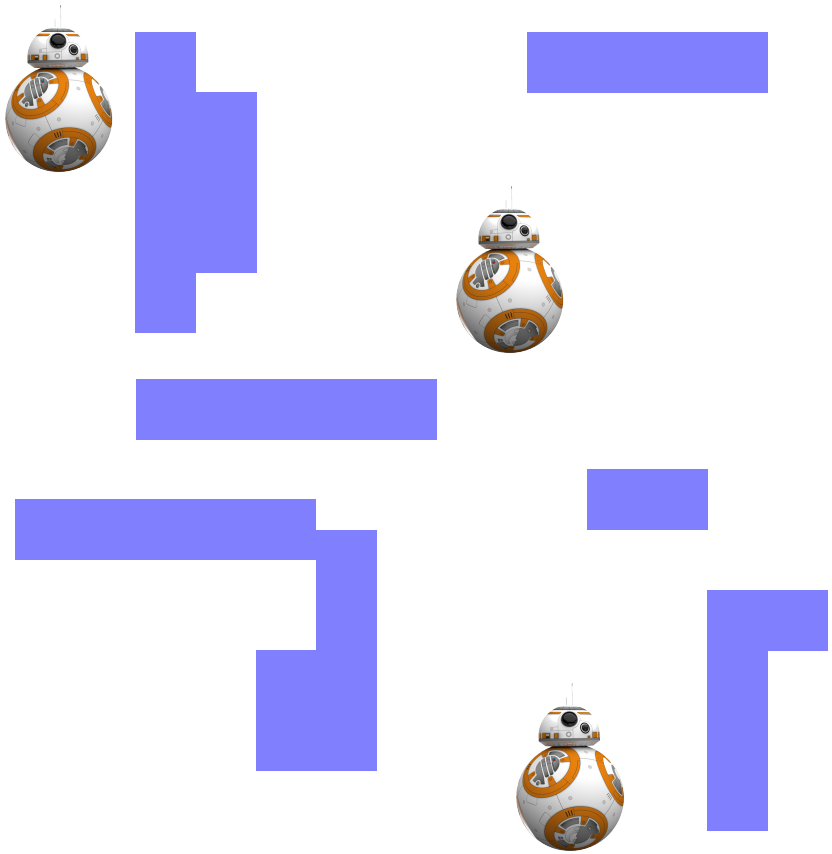


Restrictions

- Anonymous, identical
- Oblivious
- Achiral
- No message passing
- Only 1 sense: vision

A common objective

Dumb robots [SY93]



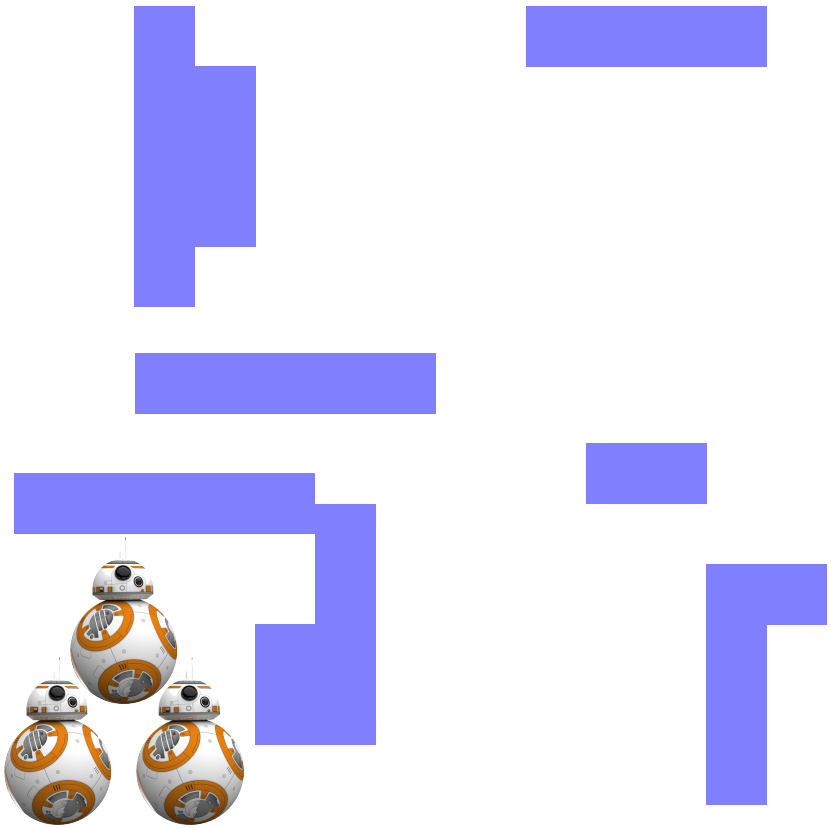
Restrictions

- Anonymous, identical
- Oblivious
- Achiral
- No message passing
- Only 1 sense: vision

A common objective

- Exploration

Dumb robots [SY93]



Restrictions

- Anonymous, identical
- Oblivious
- Achiral
- No message passing
- Only 1 sense: vision

A common objective

- Exploration
- Gathering

Verification

Properties

- Correctness
- Security and data consistency
- Fault tolerance

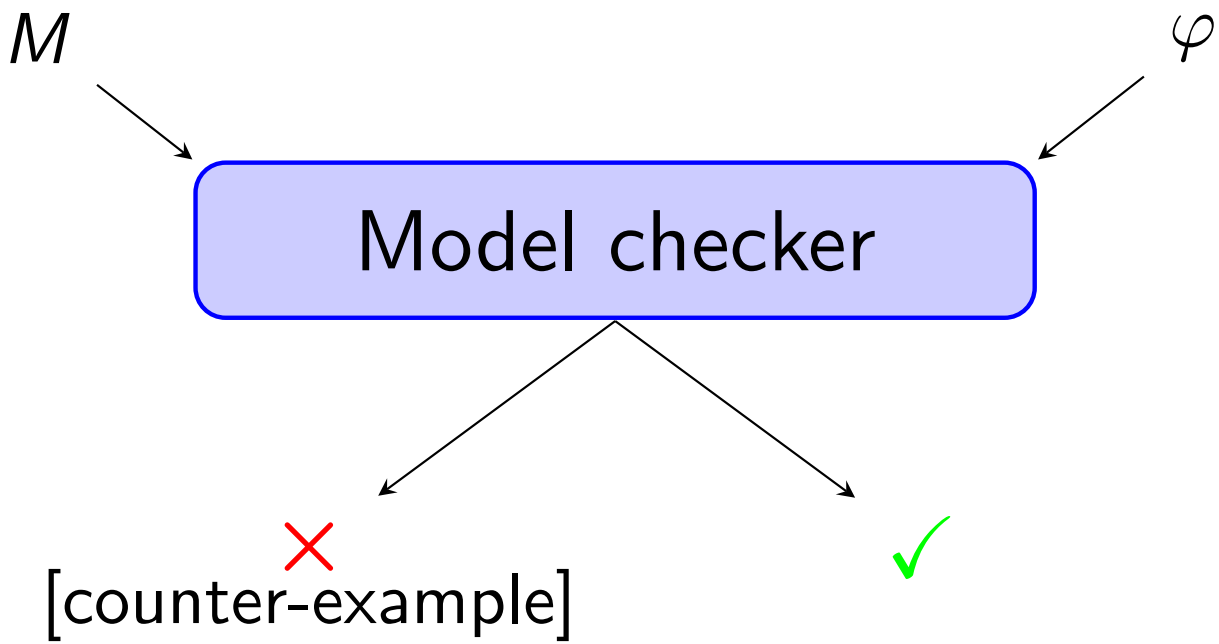
Formal methods

- Test
- Proof
- Model checking
- Synthesis

Model checking [QS81,CE82]

How to verify that a given system satisfies its specification?

System satisfies Specification



Synthesis [Church63]

How to generate a correct algorithm for a given objective?

Given φ does there exists M st. $M \models \varphi$

Distributed Synthesis is Undecidable [PR79]

Outline

A generic model for dumb robots

Verification of exploration protocols

Synthesis of gathering protocols

Conclusion

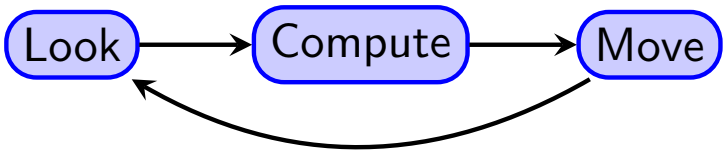
A generic model for mobile robots

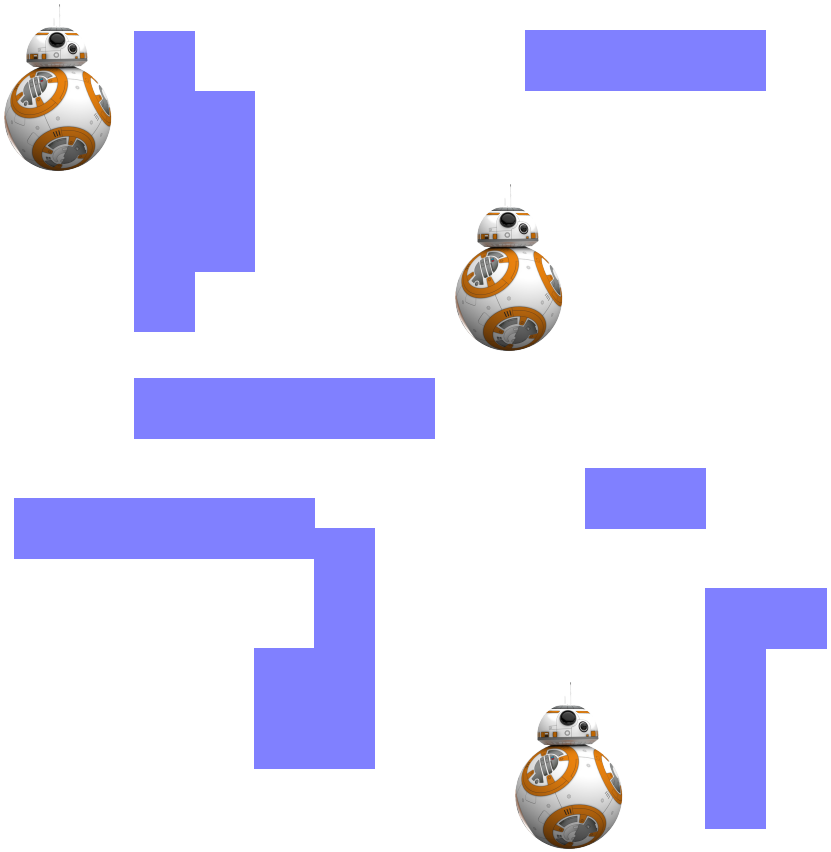
Joint work with Tali Sznajder et Yann Thierry-Mieg

Dumb robots

Restrictions

- Anonymous, identical
- Oblivious
- Achiral
- No message passing
- Only 1 sense: vision

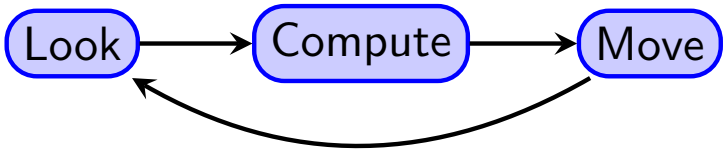


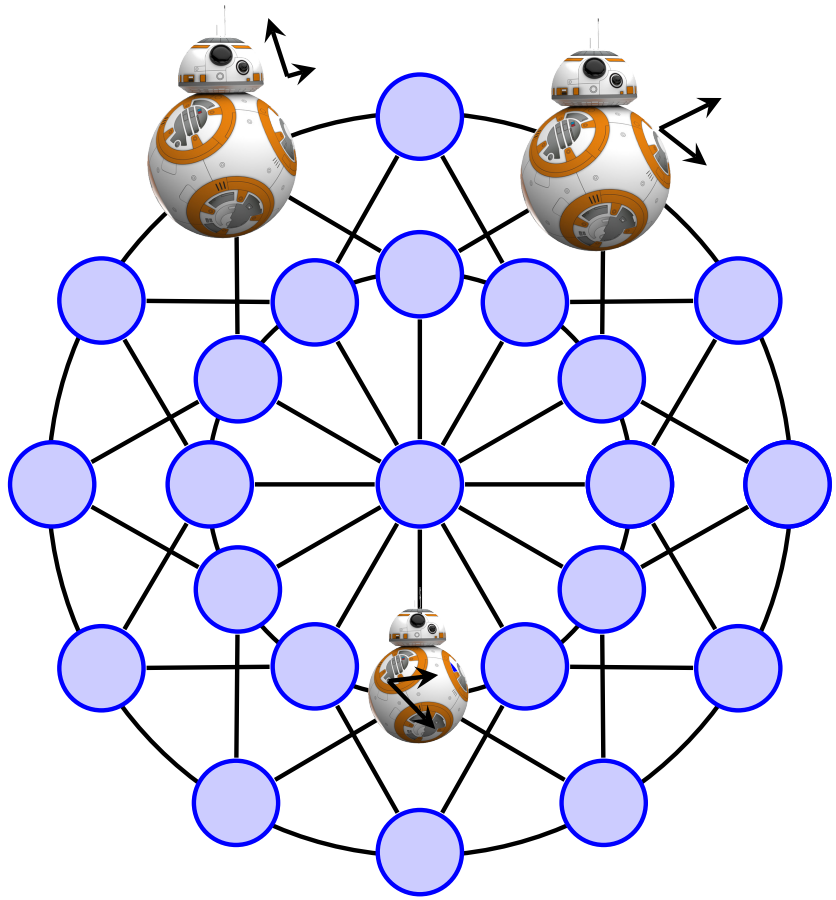


Dumb robots

Restrictions

- Anonymous, identical
- Oblivious
- Achiral
- No message passing
- Only 1 sense: vision

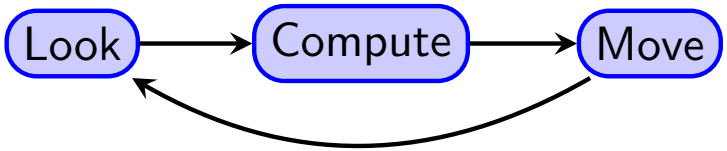


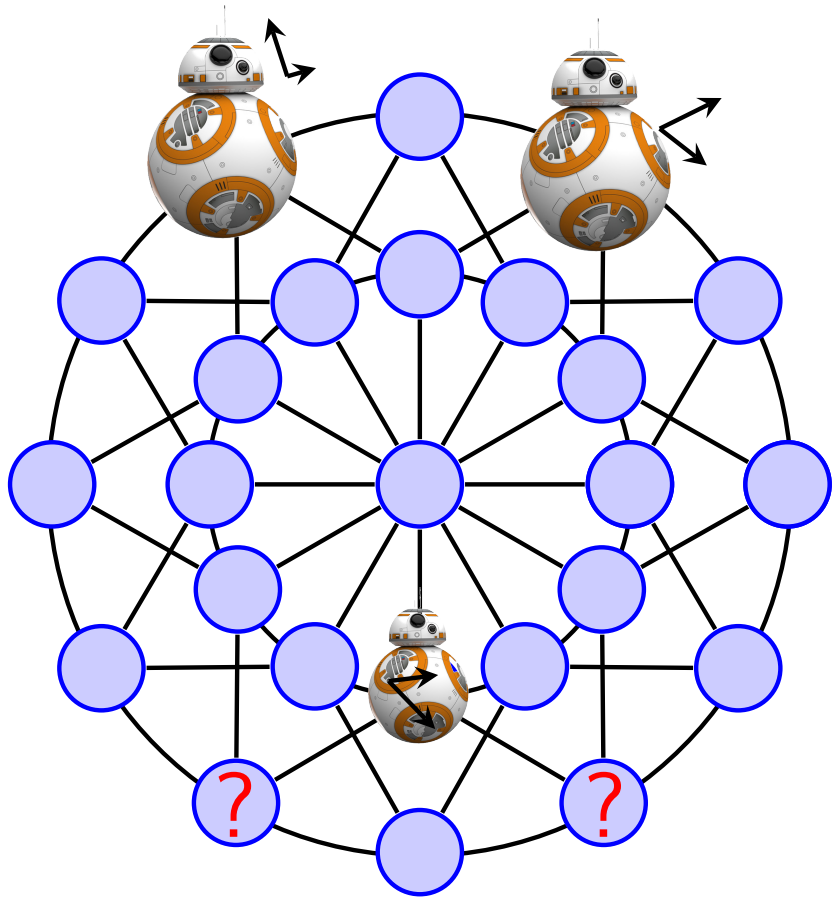


Dumb robots

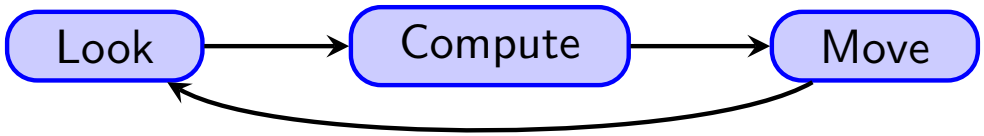
Restrictions

- Anonymous, identical
- Oblivious
- Achiral
- No message passing
- Only 1 sense: vision





Two execution models



Synchronous

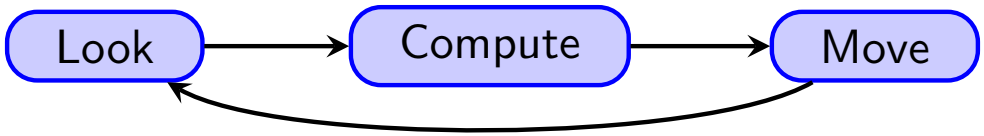
Two variants:

- Fsync
- Ssync

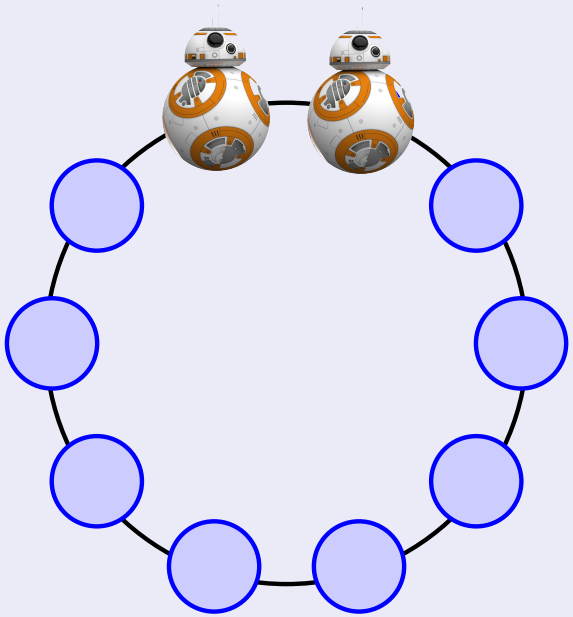
Asynchronous [Pre00]

- Async

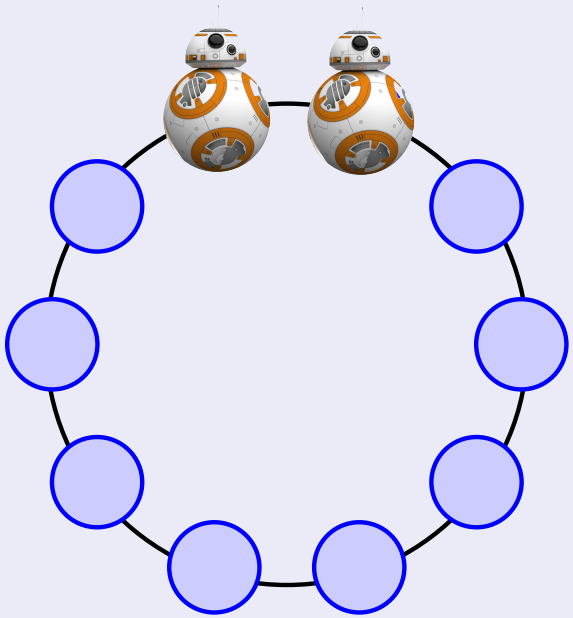
Two execution models



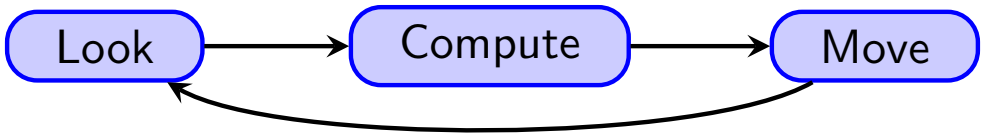
Synchronous



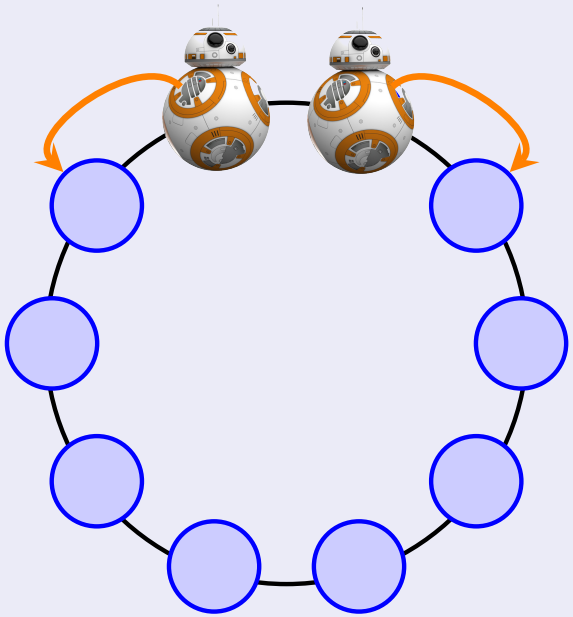
Asynchronous [Pre00]



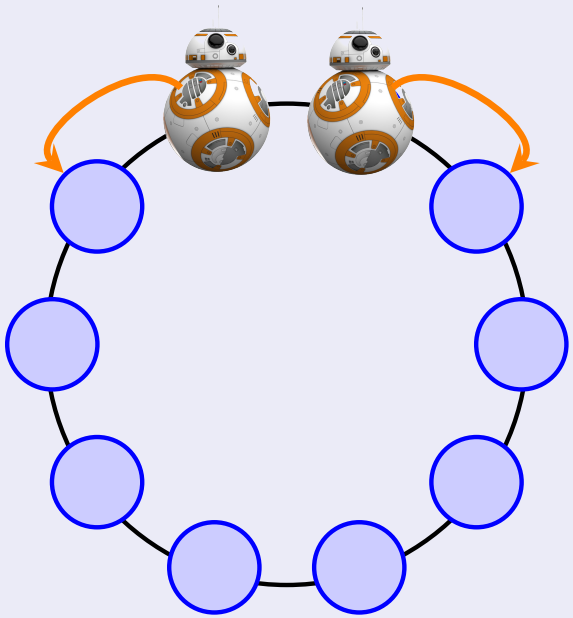
Two execution models



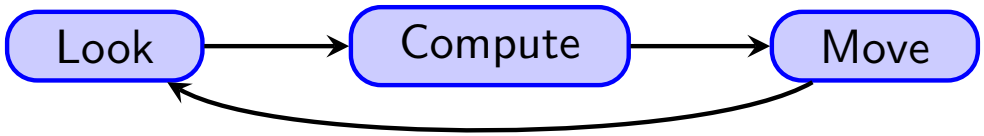
Synchronous



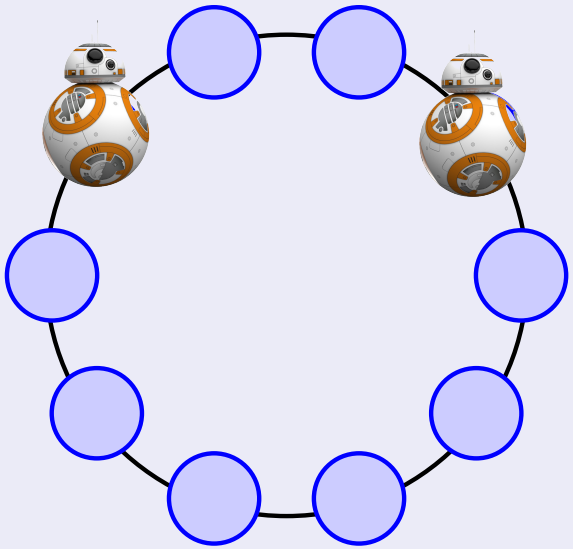
Asynchronous [Pre00]



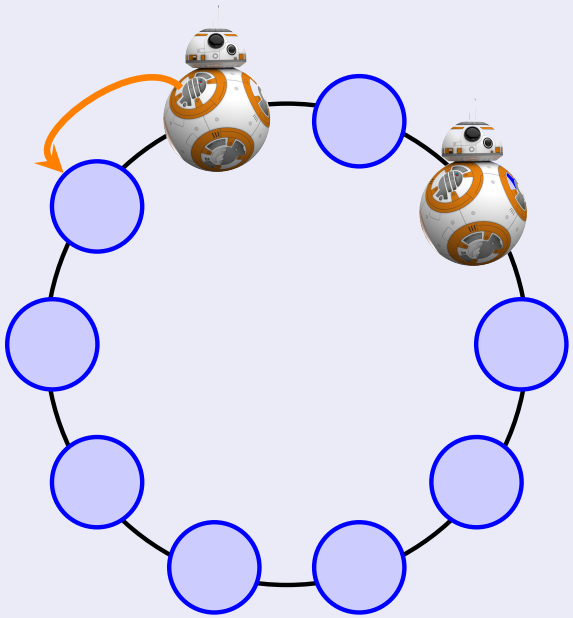
Two execution models



Synchronous



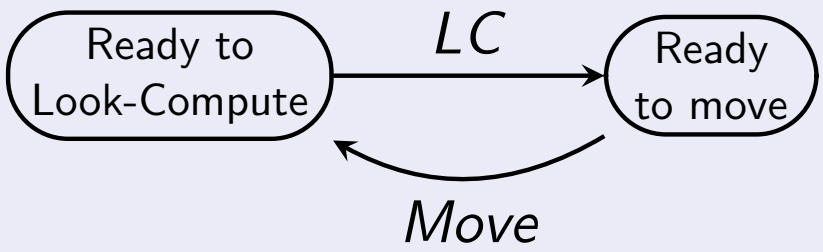
Asynchronous [Pre00]



Robot model

$Rob = \{r_1, r_2, \ldots, r_k\}$

Generic robot model

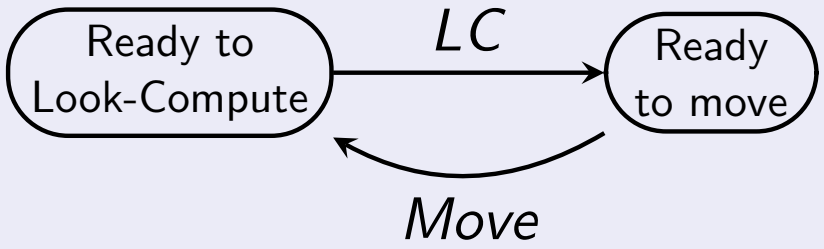


Must be instantiated according to the graph shape and the algorithm under study.

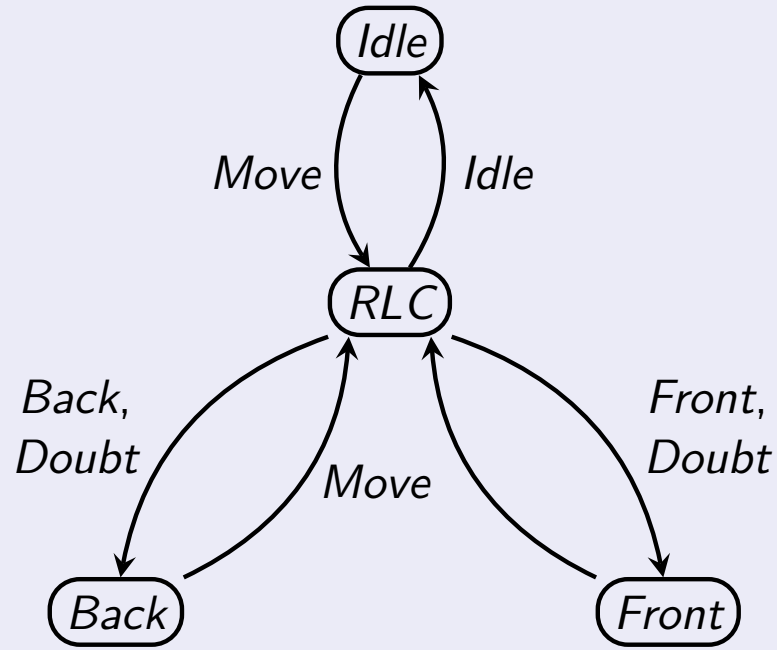
Robot model

$Rob = \{r_1, r_2, \ldots, r_k\}$

Generic robot model



Model for the ring



Must be instantiated according to the graph shape and the algorithm under study.

Scheduler

Resolves *Doubt* actions

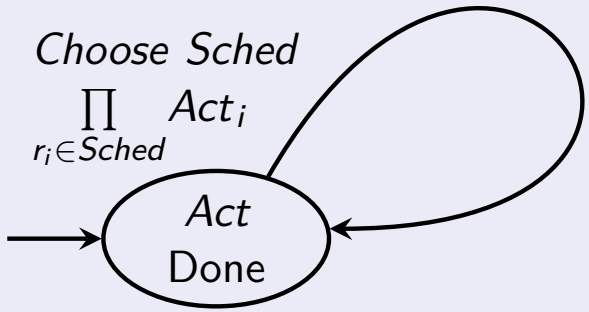
Schedules robot actions for $Sched \subseteq Rob$

Models the atomicity level (Fsync, Ssync, or Async)

Has a total knowledge of the system

Generic scheduler model

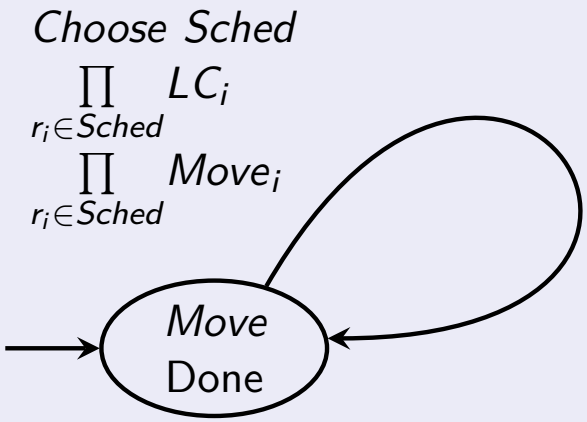
Asynchronous



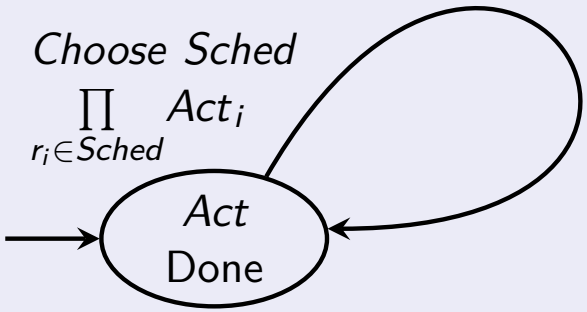
Act_i depends on robot states

Generic scheduler model

Semi-synchronous



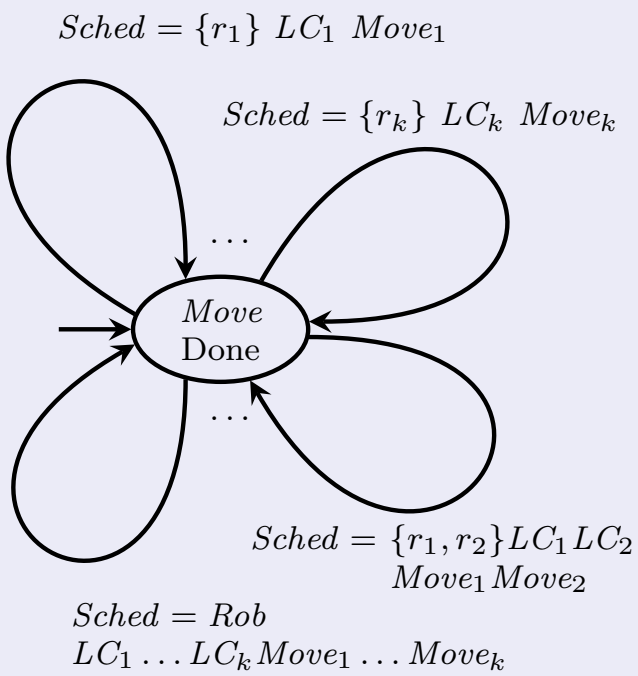
Asynchronous



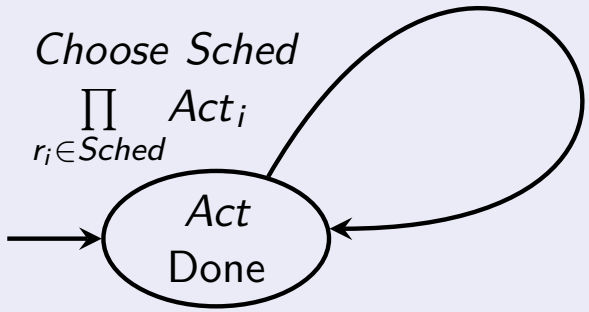
Act_i depends on robot states

Generic scheduler model

Semi-synchronous



Asynchronous



Act_i depends on robot states

Global model for the ring

Configuration $c : Rob \to Pos$

$Rob = \{r_1, r_2, \ldots, r_k\}$ $\quad$ $Pos = \{0, 1, \ldots, n-1\}$

Global model for the ring

Configuration $c : Rob \to Pos$

$Rob = \{r_1, r_2, \ldots, r_k\}$ $\quad$ $Pos = \{0, 1, \ldots, n-1\}$

A family of automata $M = (S, s_0, A, T)$

S: set of states $s = (s_1, ..., s_k, c)$ with c a configuration

$s_0 \in S$ depends on the initial configuration c_0

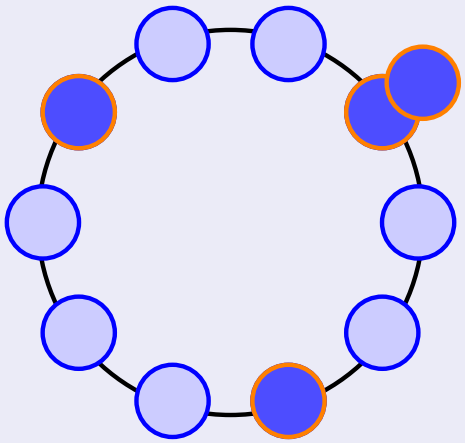
A: set of actions $A = \prod_{i \in Rob} A_i$ with $A_i = LC_i \cup Move_i$

T: set of transitions $(s_1, \ldots, s_k, c) \xrightarrow{a} (s'_1, \ldots, s'_k, c')$ with $a = (a_1, \ldots, a_k)$, where $a_i \in A_i \cup \{\varepsilon, -\}$

View

F-R-T sequence

F : free R : robot T: tower

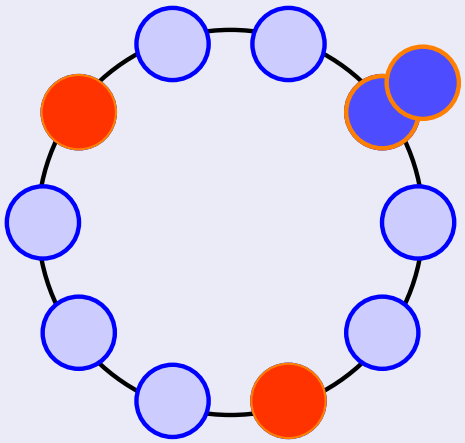


$R_1, F_2, T_2, F_2, R_1, F_3$
$T_2, F_2, R_1, F_3, R_1, F_2$

View

F-R-T sequence

F : free R : robot T: tower



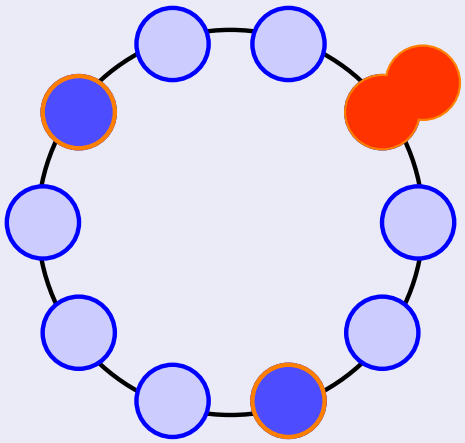
$R_1, F_2, T_2, F_2, R_1, F_3$

$T_2, F_2, R_1, F_3, R_1, F_2$

View

F-R-T sequence

F : free R : robot T: tower



$R_1, F_2, T_2, F_2, R_1, F_3$

$T_2, F_2, R_1, F_3, R_1, F_2$

Agorithm: View dependent

An algorithm = A decision function, $f : \mathit{View} \to \Delta$

$\Delta = \{\mathit{Front}, \mathit{Back}, \mathit{Idle}, \mathit{Doubt}\}$ for the ring

non-disoriented robot: *Front* or *Back* or *Idle*

disoriented robot: *Doubt* or *Idle*

Agorithm: View dependent

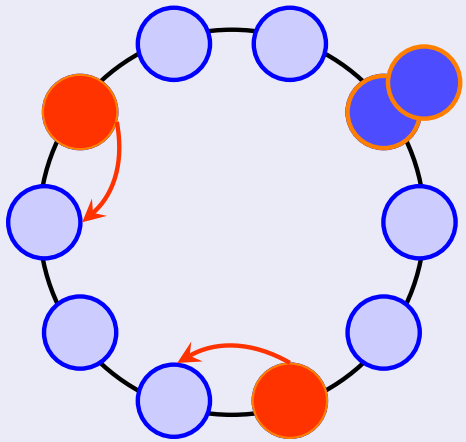
An algorithm = A decision function, $f : \mathit{View} \to \Delta$

$\Delta = \{\mathit{Front}, \mathit{Back}, \mathit{Idle}, \mathit{Doubt}\}$ for the ring

non-disoriented robot: *Front* or *Back* or *Idle*

disoriented robot: *Doubt* or *Idle*

Example



$R_1, F_2, T_2, F_2, R_1, F_3 \mapsto \mathit{Back}$

Agorithm: View dependent

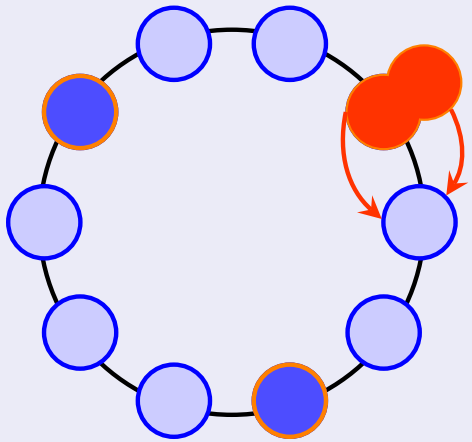
An algorithm = A decision function, $f : \mathit{View} \to \Delta$

$\Delta = \{\mathit{Front}, \mathit{Back}, \mathit{Idle}, \mathit{Doubt}\}$ for the ring

non-disoriented robot: *Front* or *Back* or *Idle*

disoriented robot: *Doubt* or *Idle*

Example



$R_1, F_2, T_2, F_2, R_1, F_3 \mapsto \mathit{Back}$

$T_2, F_2, R_1, F_3, R_1, F_2 \mapsto \mathit{Doubt}$

Agorithm: View dependent

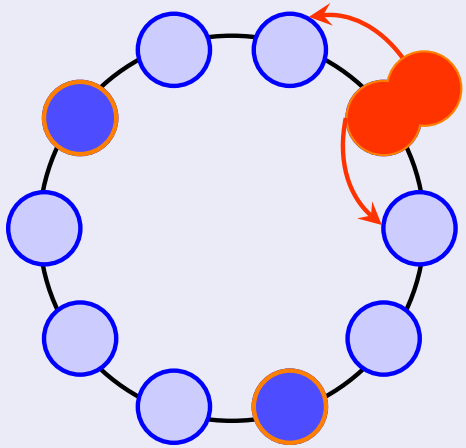
An algorithm = A decision function, $f : \mathit{View} \to \Delta$

$\Delta = \{\mathit{Front}, \mathit{Back}, \mathit{Idle}, \mathit{Doubt}\}$ for the ring

non-disoriented robot: *Front* or *Back* or *Idle*

disoriented robot: *Doubt* or *Idle*

Example



$R_1, F_2, T_2, F_2, R_1, F_3 \mapsto \mathit{Back}$

$T_2, F_2, R_1, F_3, R_1, F_2 \mapsto \mathit{Doubt}$

State of the art

Numerous robot algorithms with hand made proofs
[KKM10,Pelc11,FPS12]

Proof assistant

Certification of impossibility results [COU+15b,Aug+13]

Model checking

Small and synchronous algorithm verification [Dev+12]

Synthesis

"Brute force" synthesis [Bon+12]

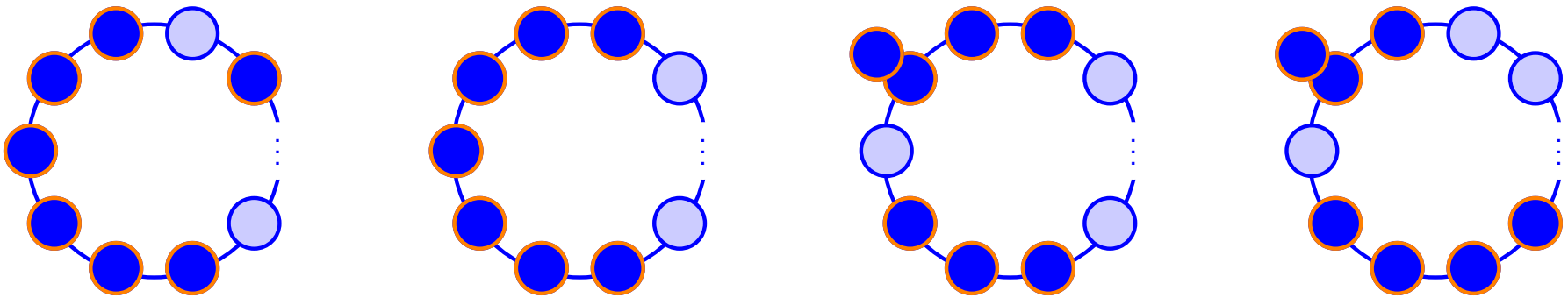
Verification of exploration protocols

Joint work with Pascal Lafourcade, Yann Thierry-Mieg and Sébastien Tixeuil

Exploration with stop [FIPS07]

Algorithm

for $k \geq 17$ robots, on a n-sized ring when n and k are coprime



SetUp TowerCreation Exploration

Formalisation

Specification in LTL

$$\textit{Exploration}: \bigwedge_{i=0}^{n-1} \Diamond \bigvee_{j=1}^{k} (c(r_j) = i)$$

$$\textit{Termination}: \bigwedge_{j=1}^{k} \Diamond\Box(\neg r_j.\textit{Front} \wedge \neg r_j.\textit{Back})$$

Implementation

15 predicates

1000 C lines of code to evaluate the predicates

26 rules

Verification of the algorithm

Verification of the SetUp phase

for $17 \leq k < 21$ robots in a ring of size and $n \leq 22$
even when n and k not coprime but without periodic initial configurations

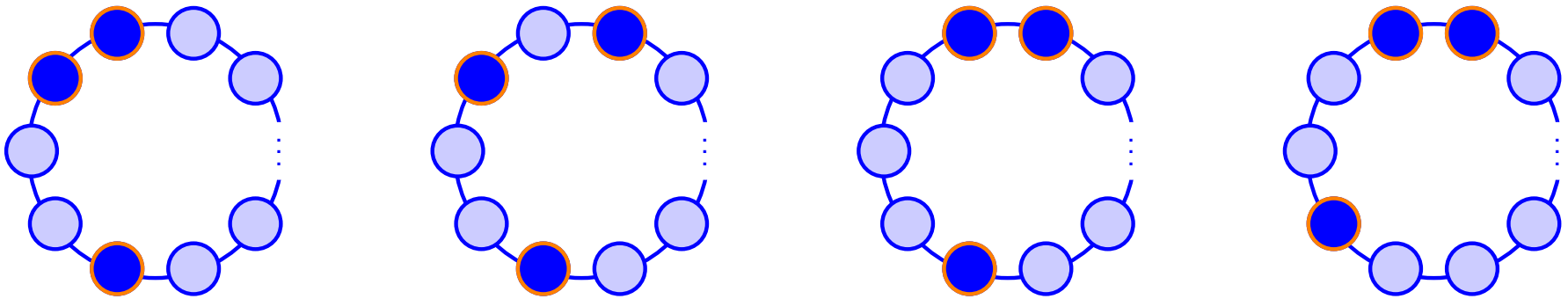
Verification of small instances

When k is even the algorithm is correct as long as $n < k + \lceil k/2 \rceil$ and $10 \leq k < 17$
When k is odd the algorithm is correct if $5 \leq k < 17$

Perpetual exploration [BMPT10]

Algorithm without collision

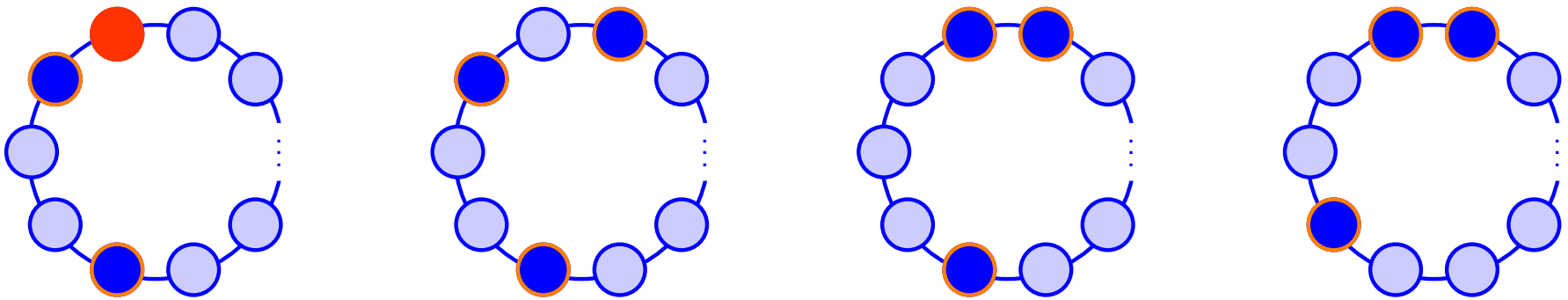
for $k = 3$ robots, on a ring of size $n \geq 10$ when n and 3 are coprime



Perpetual exploration [BMPT10]

Algorithm without collision

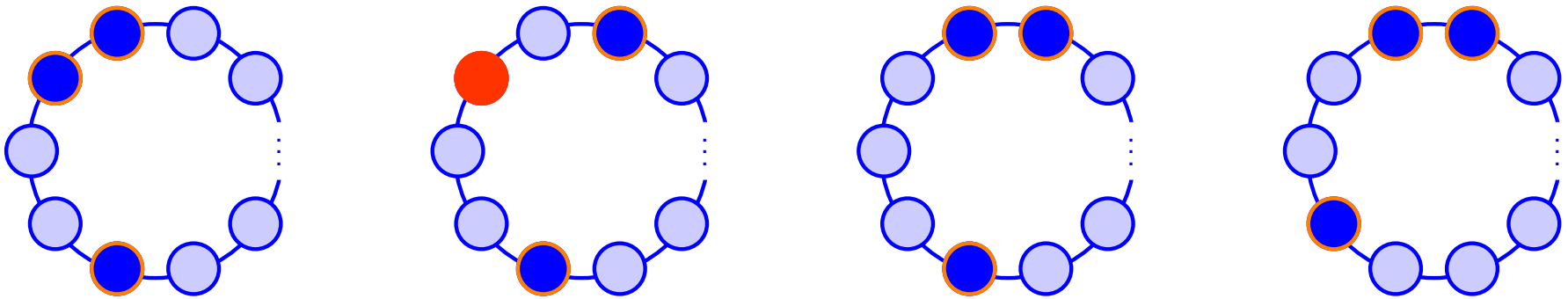
for $k = 3$ robots, on a ring of size $n \geq 10$ when n and 3 are coprime



Perpetual exploration [BMPT10]

Algorithm without collision

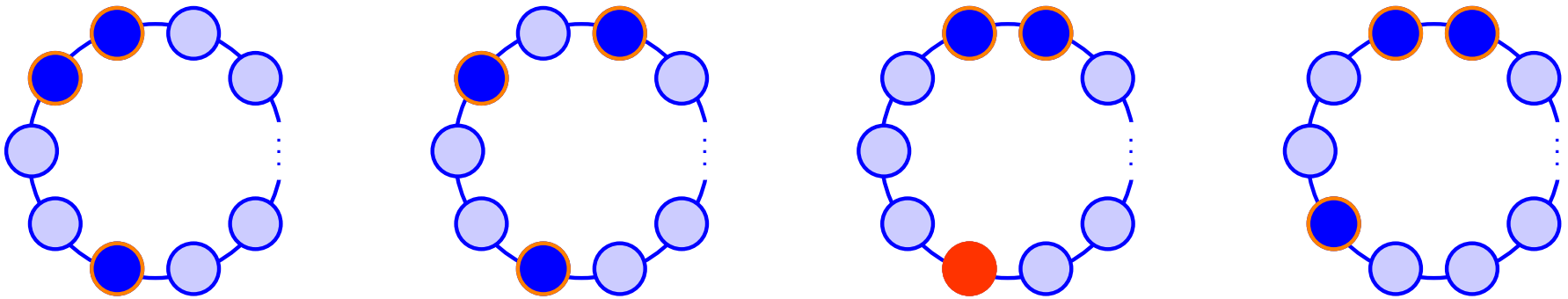
for $k = 3$ robots, on a ring of size $n \geq 10$ when n and 3 are coprime



Perpetual exploration [BMPT10]

Algorithm without collision

for $k = 3$ robots, on a ring of size $n \geq 10$ when n and 3 are coprime



Formalisation

Specification in LTL

Perpetual exploration: $\bigwedge_{i=0}^{n-1} \bigwedge_{j=1}^{k} \Box\Diamond\big(c(r_j) = i\big)$

No_collision: $\Box\Big(\bigwedge_{1\leq j<h}^{k} c(r_j) \neq c(r_h)\Big)$

No_switch:
$$\bigwedge_{i=0}^{n-1} \bigwedge_{j=1}^{k} \bigwedge_{h=1}^{k} \neg\Diamond\Big(c(r_j) = i \ \wedge\ c(r_h) = i+1 \ \wedge\ r_j.Front \ \wedge\ r_h.Back\Big)$$

Verification of the algorithm

Model checking

correct for a ring of size 10 in Fsync

correct for a ring of size 10 in Ssync

incorrect for a ring of size 10 in Async

Model checking for the corrected algorithm

For ring size up to 16 nodes

Proof

Inductive proof: base case = model checking

Summary

Flocchini: exploration with stop

- Formalisation of the algorithm
- Refined bounds

Min-Algorithm: perpetual exploration [AlgoTel'13]

- A counter example
- Correction + induction proof

[submitted to DC]

Synthesis of gathering protocols

Joint work with Tali Sznajder and Sébastien Tixeuil

Synthesis

Objective

A gathering algorithm

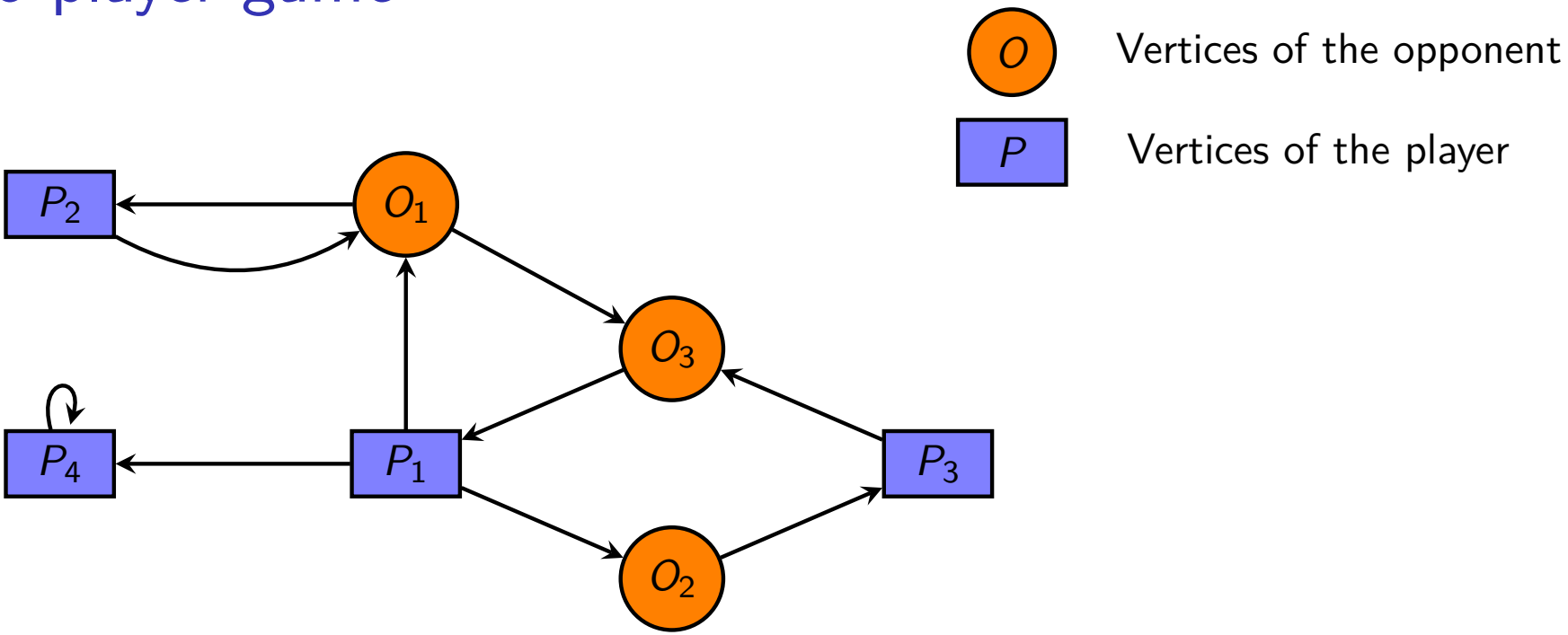
A two-player reachability game

Player: coalition of robots

Opponent: scheduler

Find a winning strategy for Player to achieve gathering

Two-player game



An Arena $\mathcal{A} = (V, E)$

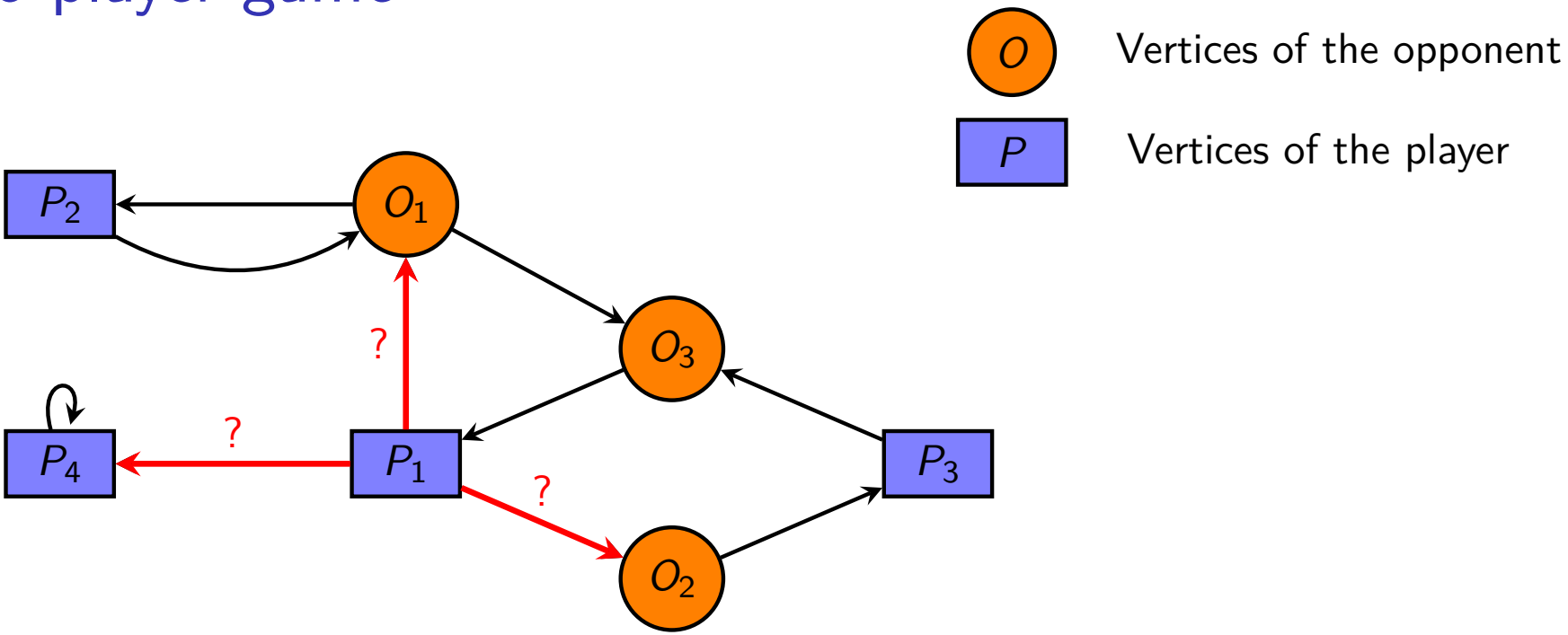
$V = V_P \uplus V_O$

$E \subseteq V \times V$

A Play in $\mathcal{A}$, $\pi \in V^\infty$

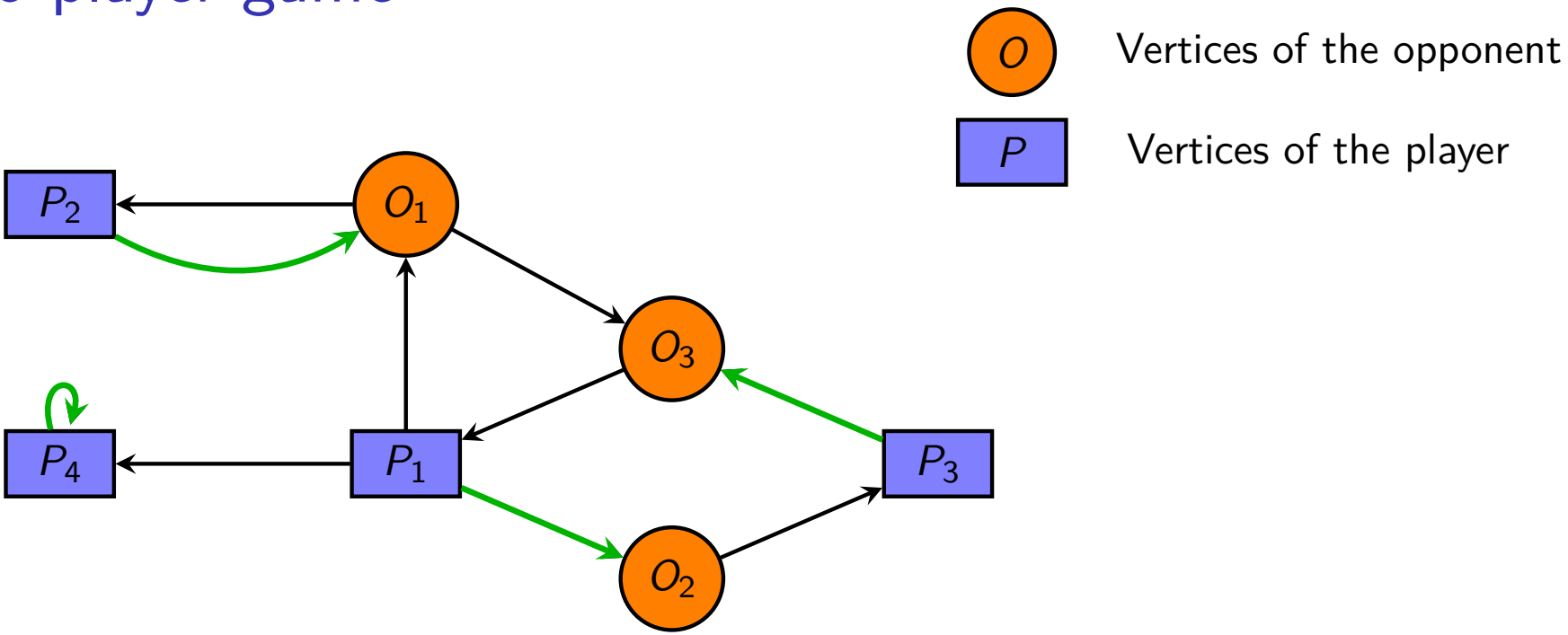
$\pi = O_1 P_2 O_1 O_3 P_1 P_4 \cdots$

Two-player game



A Strategy for the player

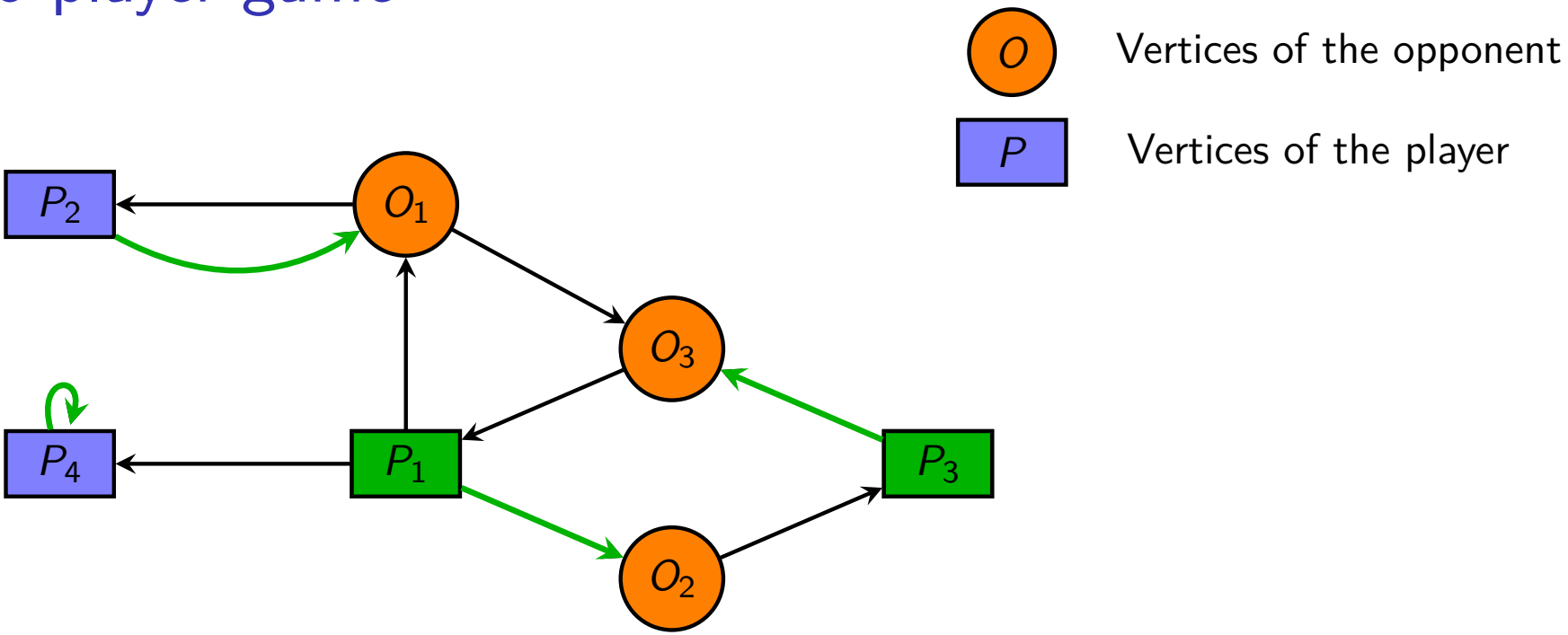
Two-player game



A Strategy for the player

A memoryless strategy:
P_1 leads to O_2, P_2 leads to O_1, P_3 leads to O_3, P_4 leads to P_4.

Two-player game



Winning strategy

Is this strategy winning?

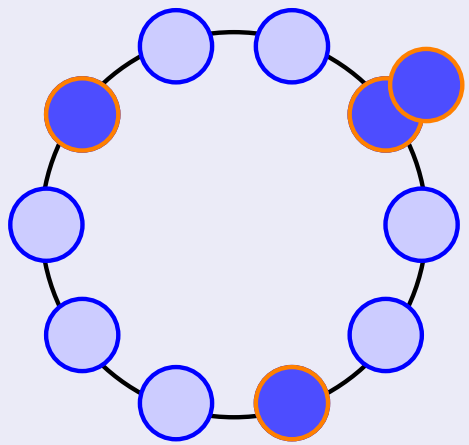
Winning condition $\subseteq V^\infty$

$Reach(P_3)$

Strategy $\rightarrow$ robot algorithm

Free nodes representation

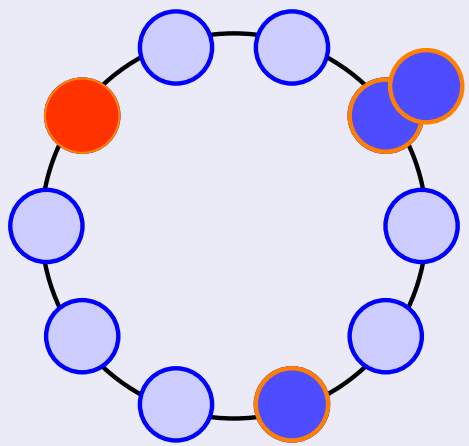
$F = (f_1, \cdots, f_k)$, such that $\Sigma_{i=1}^{k} f_i = n - k$, and $f_i \in \mathbb{N} \cup \{-1\}$



Strategy $\rightarrow$ robot algorithm

Free nodes representation

$F = (f_1, \cdots, f_k)$, such that $\Sigma_{i=1}^{k} f_i = n - k$, and $f_i \in \mathbb{N} \cup \{-1\}$

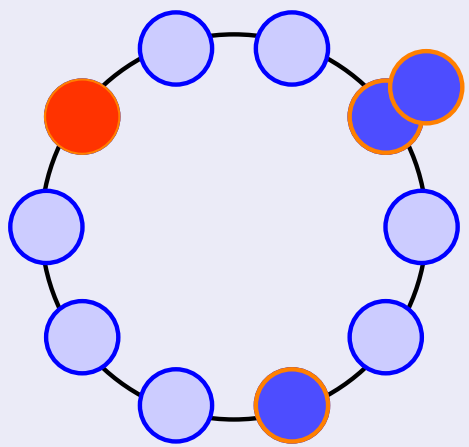


$(3, 2, -1, 2)$

Strategy $\rightarrow$ robot algorithm

Free nodes representation

$F = (f_1, \cdots, f_k)$, such that $\Sigma_{i=1}^{k} f_i = n - k$, and $f_i \in \mathbb{N} \cup \{-1\}$



$(3, 2, -1, 2)$

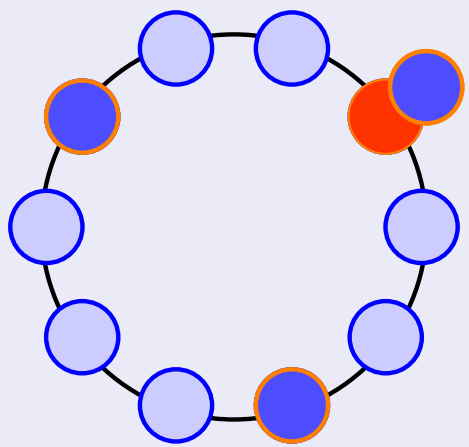
$(2, -1, 2, 3)$

$\sim$

Strategy $\rightarrow$ robot algorithm

Free nodes representation

$F = (f_1, \cdots, f_k)$, such that $\Sigma_{i=1}^{k} f_i = n - k$, and $f_i \in \mathbb{N} \cup \{-1\}$



$(3, 2, -1, 2)$

$(2, -1, 2, 3)$

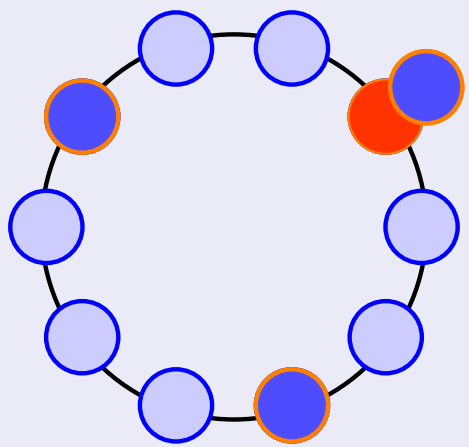
$(-1, 2, 3, 2)$

$\sim$

Strategy $\rightarrow$ robot algorithm

Free nodes representation

$F = (f_1, \cdots, f_k)$, such that $\Sigma_{i=1}^{k} f_i = n - k$, and $f_i \in \mathbb{N} \cup \{-1\}$



$(3, 2, -1, 2)$ $\sim$

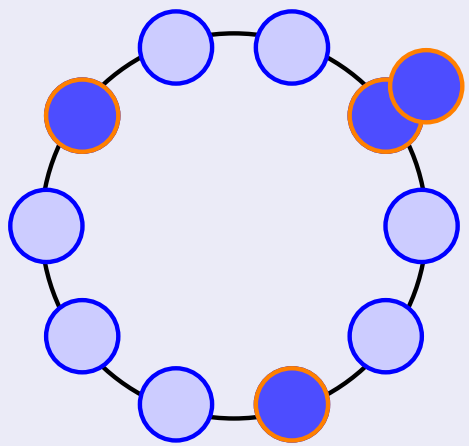
$(2, -1, 2, 3)$ $\circlearrowleft$

$(-1, 2, 3, 2)$

Strategy $\rightarrow$ robot algorithm

Free nodes representation

$F = (f_1, \cdots, f_k)$, such that $\Sigma_{i=1}^{k} f_i = n - k$, and $f_i \in \mathbb{N} \cup \{-1\}$



$$(3, 2, -1, 2) \sim$$
$$(2, -1, 2, 3) \circlearrowleft$$
$$(-1, 2, 3, 2)$$

Equivalence class $\equiv$

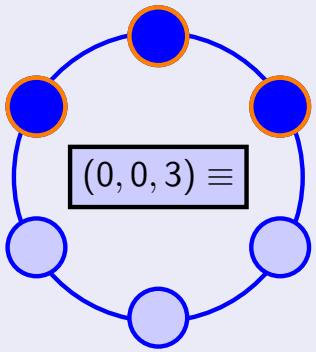
$\equiv \stackrel{\text{def}}{=} (\circlearrowleft \cup \sim)^*$

Synchronous arena

The Arena, $\mathcal{A}_{\text{gather}} = (V_P \uplus V_O, E)$

$$V_P = (\mathcal{F}/\equiv)$$

An example for 3 robots in a 6 nodes ring



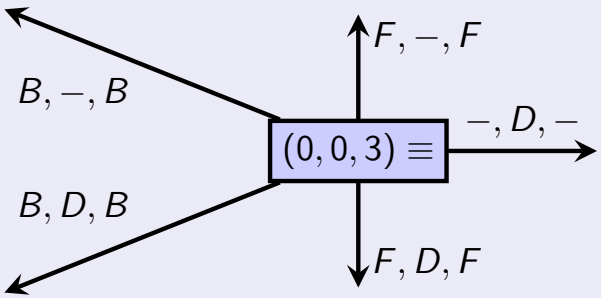
Synchronous arena

The Arena, $\mathcal{A}_{\text{gather}} = (V_P \uplus V_O, E)$

$$V_P = (\mathcal{F}/\equiv)$$

$$E \subseteq (V_P \times V_O) \cup$$

An example for 3 robots in a 6 nodes ring



Synchronous arena

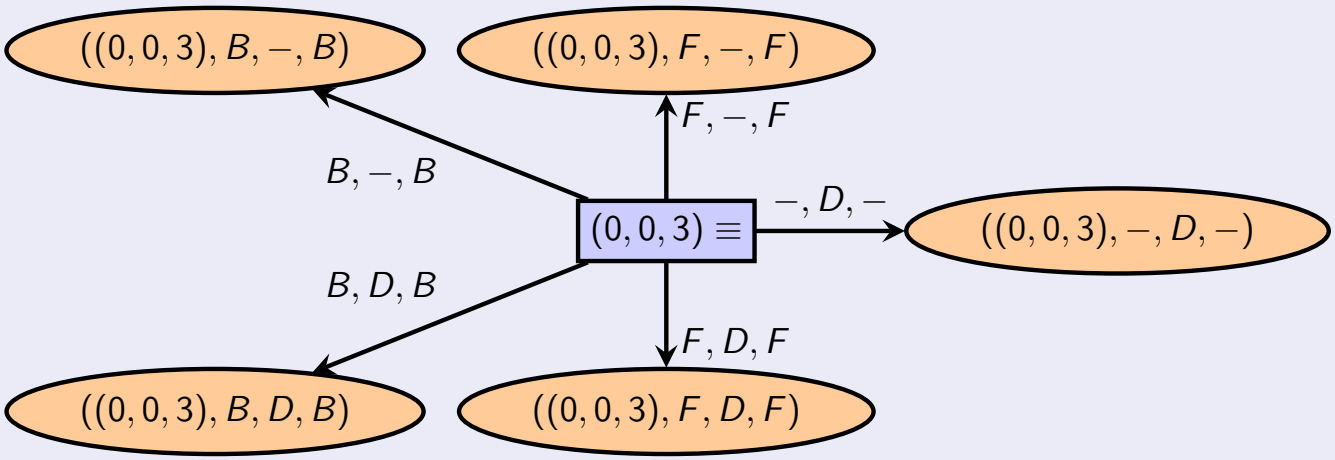
The Arena, $\mathcal{A}_{\text{gather}} = (V_P \uplus V_O, E)$

$$V_P = (\mathcal{F}/\equiv)$$

$$V_0 = \mathcal{F} \times \Delta^k$$

$$E \subseteq (V_P \times V_O) \cup$$

An example for 3 robots in a 6 nodes ring

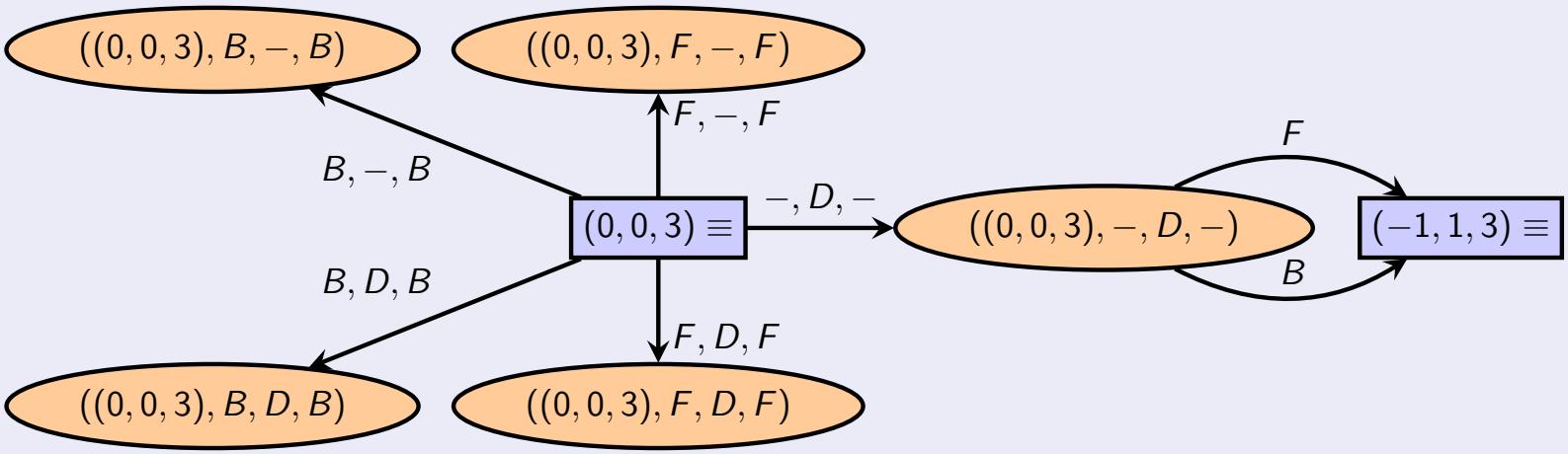


Synchronous arena

The Arena, $\mathcal{A}_{\text{gather}} = (V_P \uplus V_O, E)$

$$V_P = (\mathcal{F}/\equiv)$$
$$V_0 = \mathcal{F} \times \Delta^k$$
$$E \subseteq (V_P \times V_O) \cup (V_O \times V_P)$$

An example for 3 robots in a 6 nodes ring



Synchronous arena

The Arena, $\mathcal{A}_{\text{gather}} = (V_P \uplus V_O, E)$

$$V_P = (\mathcal{F}/\equiv)$$

$$V_0 = \mathcal{F} \times \Delta^k$$

$$E \subseteq (V_P \times V_O) \cup (V_O \times V_P)$$

Synchronous arena

The Arena, $\mathcal{A}_{\text{gather}} = (V_P \uplus V_O, E)$

$$V_P = (\mathcal{F}/\equiv)$$
$$V_0 = \mathcal{F} \times \Delta^k$$
$$E \subseteq (V_P \times V_O) \cup (V_O \times V_P)$$

Theorem

There exists an algorithm which achieves gathering if and only if there exists a memoryless winning strategy for the reachability game $\mathcal{A}_{\text{gather}}$ with winning condition $\text{Reach}(\{(-1, \ldots, -1, n-1) \equiv\})$.

Results for synchronous robots [SSS'14,AlgoTel'15]

Impossibility results

For some initial configurations

The best algorithm

For 3 robots in ring of size in $3..16, 100$

In $\lceil n/2 \rceil$ rounds

A parameterized algorithm

Pattern finding

Correctness by an inductive proof

Conclusion

A model for robot algorithm

Verification [AlgoTel'13] [DC]

Model generator for robot algorithms on a ring
Verification and correctness of existing algorithms

Synthesis [SSS'14,AlgoTel'15]

Synthesis and correctness of an algorithm for 3 synchronous robots
An algorithm to synthesize robot protocols in the Async model

Perspectives

Asynchronous Synthesis

For 4 robots

For k robots

Verification of parametrized systems

Ongoing work: semi automatic tool combining abstraction and model checking+ induction

Proof: Extension of the Coq based Pactole framework

For asynchronous robots

Decidability questions

For the verification and synthesis of parameterized robot algorithms