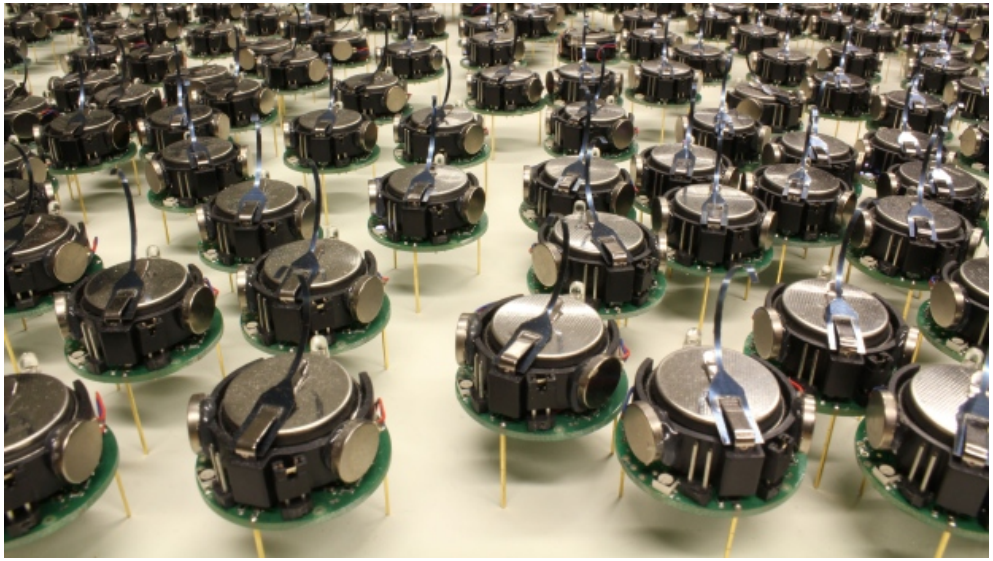
Safety and Versatility

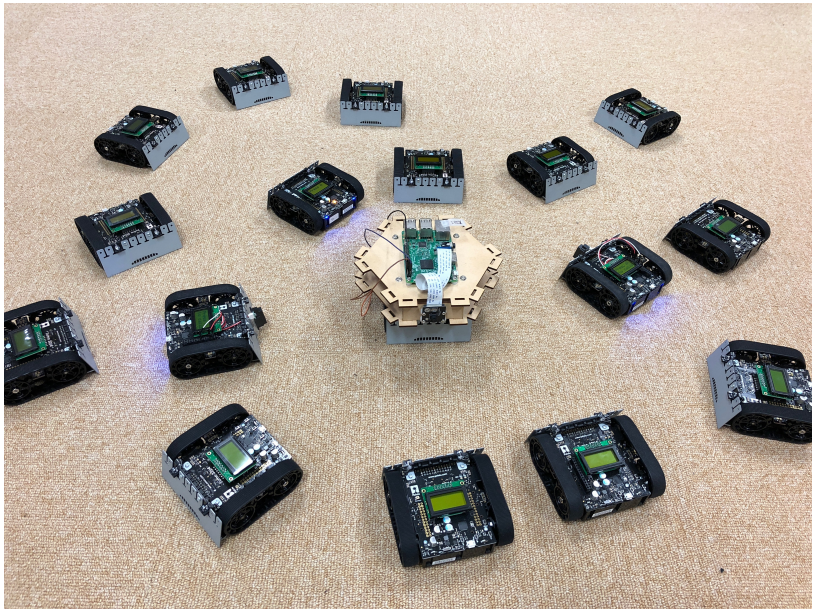
Verifying Swarms of Mobile Robots

Xavier Urbain UCBL-1–LIRIS

Autonomous Mobile Robots



Autonomous Mobile Robots

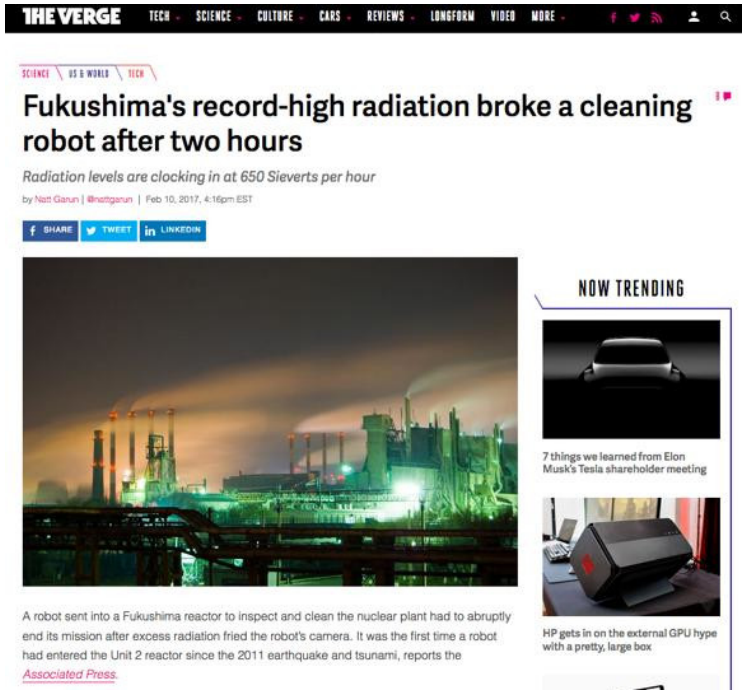


Autonomous Mobile Robots

cooperation

- Dynamic networks
- Exploration/Patrol
- Search and Rescue
- . . .

Environment: hostile?



Autonomous Mobile Robots

cooperation

- Dynamic networks
- Exploration/Patrol
- Search and Rescue
- . . .

EnvironmentEnvironment: hostile?

- Cheap
- Protocols Robust

⇝ Assumptions low

⇝ Expectations high

Task: critical!

the userthe user and

the taskthe task can loose agents

capabilities, silence. . .

Guarantee through **formal methods**

Distributed Computing

certification

Subtle variations + informal reasoning ⇝ source of errors

Protocols incorrect wrt specifications still found

Recent, in expansion + critical app. ⇝ need for sound foundations

Formal Methods

- Model-checking: reachability LTL [Défago. . . , Bérard. . . , Devisme. . .]
  + automation − instances (limited, discrete)
- Synthesis [Bonnet. . . , Millet. . .]
- Formal proof : mechanically assisted Coq, Isabelle
  − expertise + scalable, generic

⇝ 2 phases spec/proof: emphasis on easy specification

Mobile Robots

needs

From a computer science point of view:

- Model characterized clearly, towards "realistic" variants
- Tools (theory+software) for formal verification

Suzuki & Yamashita 1999

Robots move in **space** according to their **perception**

Following a cycle: 

Robots autonomous $\rightsquigarrow$ cooperate in a task

Without any explicit communication channel

$\rightsquigarrow$ many variants...

Suzuki & Yamashita 1999

Robots move in **space** according to their **perception**

Characteristics of space

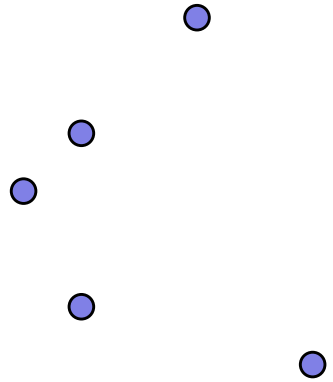
- Topology?
- Discrete/Continuous ?
- Bounded/infinite?
- ...

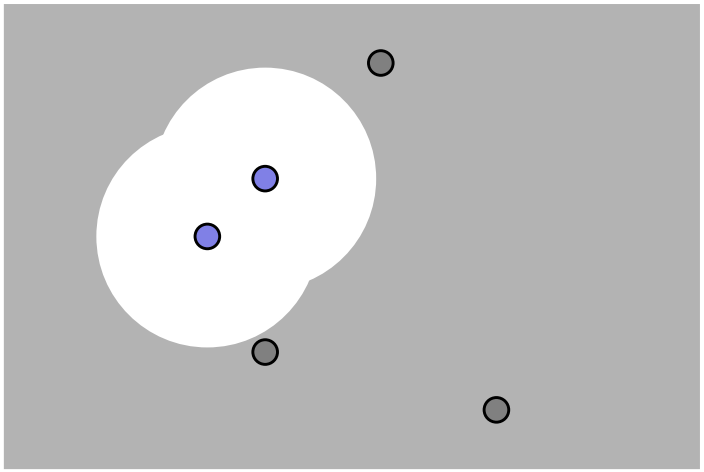
Suzuki & Yamashita 1999

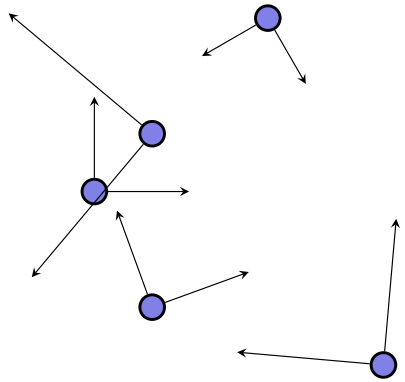
Robots move in **space** according to their **perception**

Capabilities (robots/sensors)

- Memory/Oblivious? (no memory of previous cycle)
- Vision limited/global?
- Orientation share/chirality? left/right?
- Perception? names? multiplicity? ("3 robots here"≠"some here")
- ...







Self-visible?

Suzuki & Yamashita 1999

Robots move in space according to their perception

Synchronization and **movement**

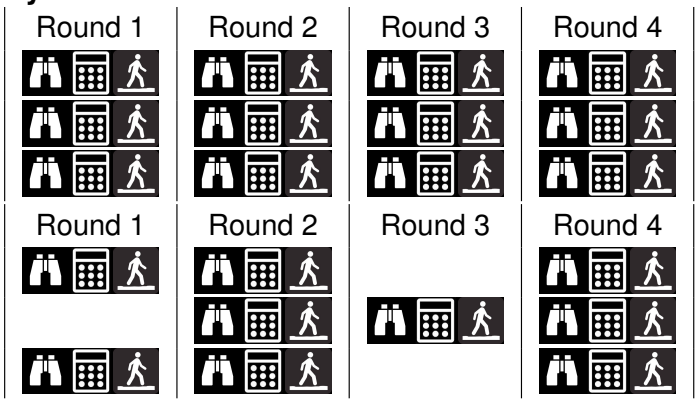
- Model of synchro?
- Trajectory?
- Speed?
- . . .

\+ (Byzantine) faults!

Suzuki & Yamashita 1999

Robots move in **space** according to their **perception**

Synchronization and **movement**



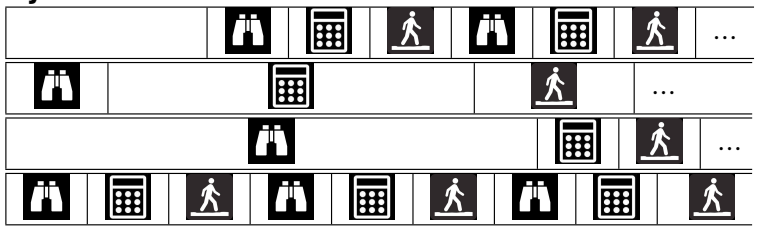
FSYNC

SSYNC

Suzuki & Yamashita 1999

Robots move in **space** according to their **perception**

Synchronization and **movement**



ASYNC

$\rightsquigarrow$ outdated perceptions...

ASYNC $\subseteq$ SSYNC $\subset$ ASYNC$^{O(1)}$

Core

```
Definition round r da config : configuration :=
 fun id ⇒ (* for a given robot, the new configuration *)
  let state := config id in
  if da.(activate) id then match id with (* activated *)
    match id with
     | Byz b ⇒ da.(relocate_byz) config b (* by demon *)
     | Good g ⇒       (* change the frame of reference *)
       let frame_choice := ... in let new_frame := ... in
       let local_conf := ... in   let local_st := ...  in
       (* compute the observation and apply r *)
       let obs := obs_from_config local_conf local_st in
       let loc_robot_dec := r obs in
       (* demon choice on how to update state *)
       let choice := da.(choose_update)... loc_robot_dec in
       (* actual update and return to the global frame *)
  else inactive config id (da.(choose_inactive) config id).
```


Swarms in Continuous Space

Questions:

- What is possible?
- How it is possible?
- Is that correct?

Popular Problems

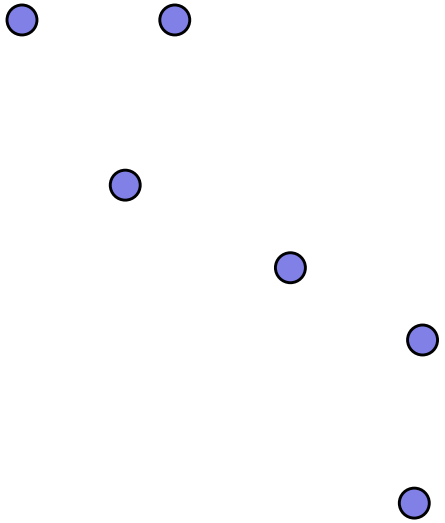
- Probe (patrol/exploration)
- Pursuit (flock/school)
- Pattern formation

Swarms in Continuous Space

pattern

Origin: any conf. (scattered)

Goal: Arbitrary pattern

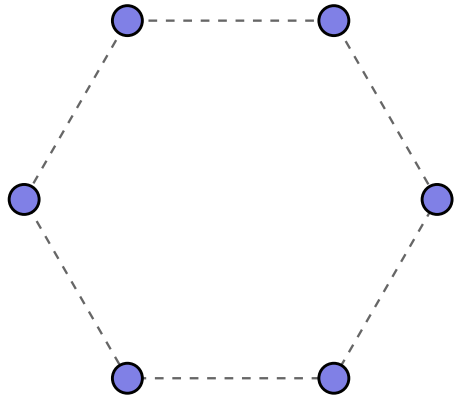


Swarms in Continuous Space

pattern

Origin: any conf. (scattered)

Goal: Arbitrary pattern



Swarms in Continuous Space

pattern

Origin: any conf. (scattered)

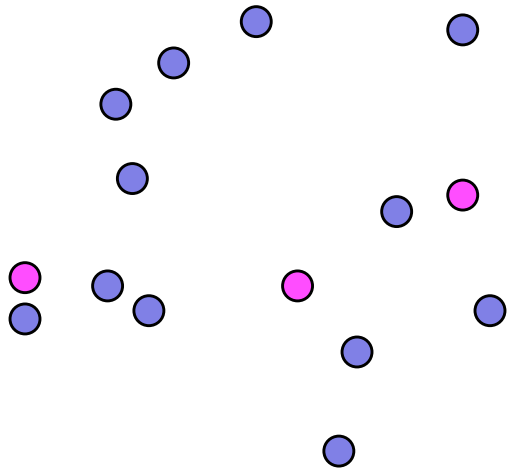
Goal: Arbitrary pattern

Subcases: benchmarks

- Uniform circle/polygons, *regular*-ish shapes. . .
- Convergence/Gathering

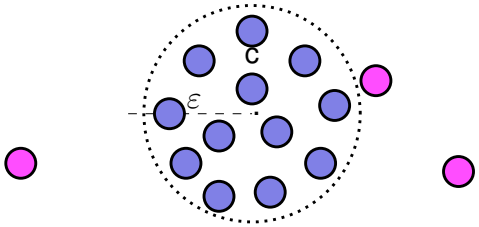
Swarms in Continuous Space

initial



Swarms in Continuous Space

convergence

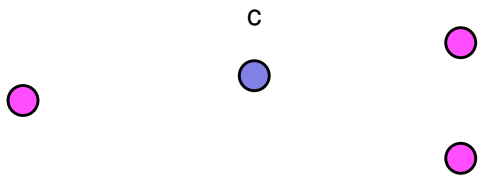


- Forever within ε from c (coinductive)
- Eventually there (inductive)

Convergence: $\exists c, \forall \varepsilon \ldots$

Swarms in Continuous Space

gathering



- Forever at c (coinductive)
- Eventually there (inductive)

Gathering: $\exists c, \ldots$

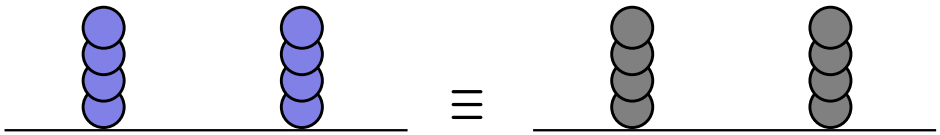
Formalisation

SSYNC/mult./rigid

```
Definition Gather (pt: Location.t) (e: execution) :=
  Stream.forever (Stream.instant (gathered_at pt)) e.
Definition WillGather (pt: Location.t) (e: execution) :=
  Stream.eventually (fun e ⇒ ∃ pt, Gather pt e) e.

Definition FullSolGathering (r: robogram) (d: demon) :=
  ∀ conf, WillGather (execute r d conf).

Hypothesis even_nG : Nat.Even N.nG.
Hypothesis nG_non_0 : N.nG ≠ 0.
Variable r : robogram.

Theorem noGathering_Fair:
∀ config, invalid config →
∃ d, Fair d ∧ ¬WillGather (execute r d config).
```


Example of Impossibility

gathering even

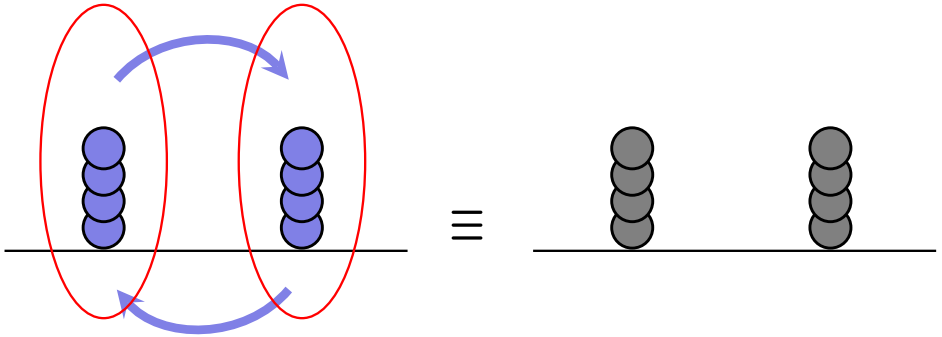


Example of Impossibility

gathering even

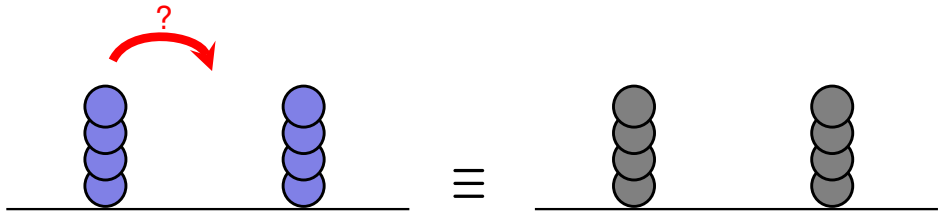
Example of Impossibility

gathering even



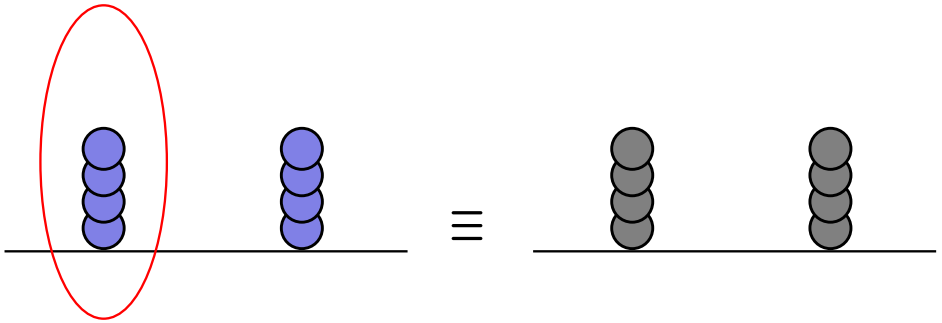
Example of Impossibility

gathering even



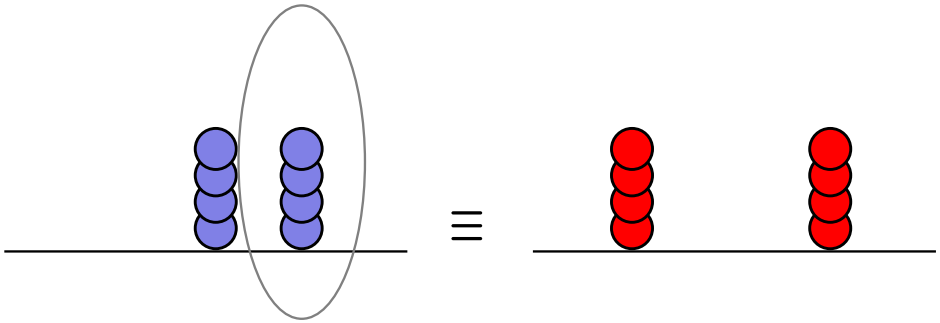
Example of Impossibility

gathering even



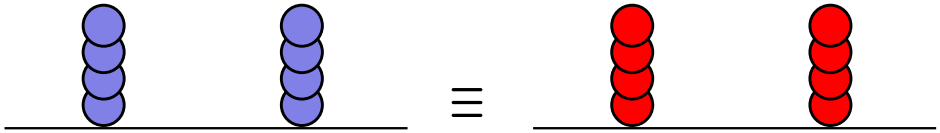
Example of Impossibility

gathering even



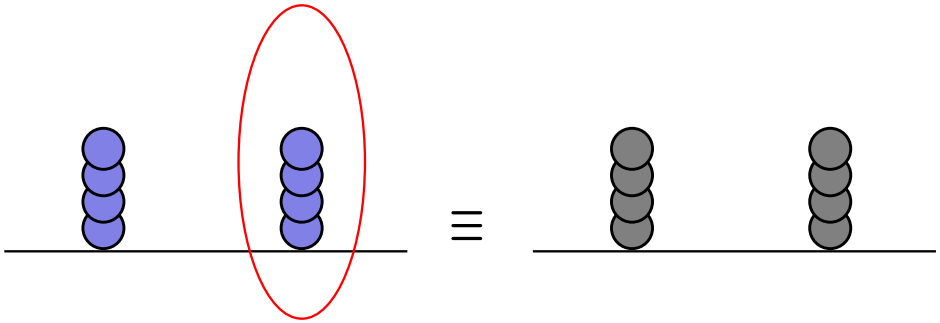
Example of Impossibility

gathering even



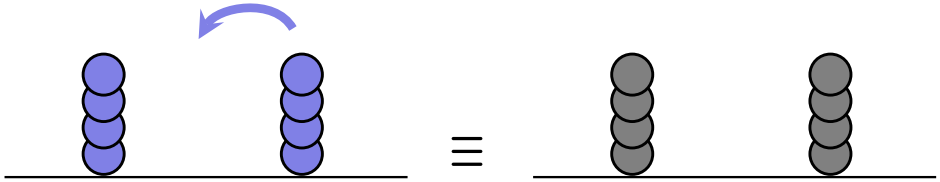
Example of Impossibility

gathering even



Example of Impossibility

gathering even



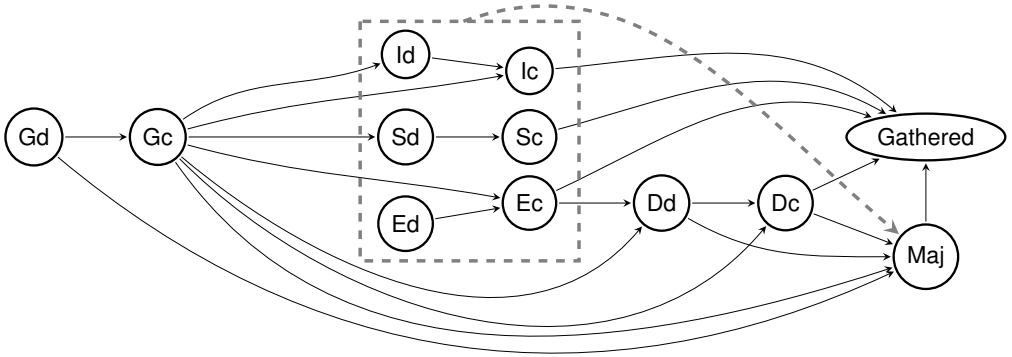
Formalisation

SSYNC/mult./rigid

```
Definition Gather (pt: Location.t) (e: execution) :=
  Stream.forever (Stream.instant (gathered_at pt)) e.
Definition WillGather (pt: Location.t) (e: execution) :=
  Stream.eventually (fun e ⇒ ∃ pt, Gather pt e) e.

Definition ValidSolGathering (r: robogram) (d: demon) :=
  ∀ conf, ¬invalid conf → WillGather (execute r d conf).

Definition gatherR2...                    (* first solution *)

Theorem Gathering_in_R2 : ∀ d, Fair d →
  ValidSolGathering gatherR2 d.          (* proof 25 lines *)
                                         (* total 3000 lines *)
```


GatherR2

SSYNC/mult./rigid



Mobile Robots

needs

From a computer science point of view:

- Model characterized clearly, towards "realistic" variants
- Tools (theory+software) for formal verification

In practice: problem oriented

Academic/Fundamental problems Gathering...

⇝ contradiction!

How to obtain a correct protocol?

Context: Suzuki & Yamashita Task: Rescue/Lifeline

Keeping in touch

task

Keeping in touch

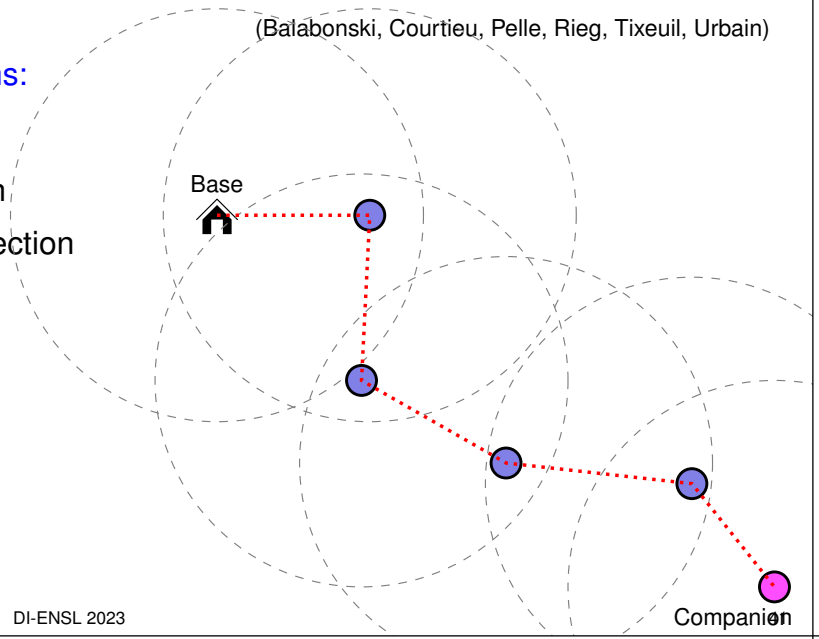
task

(Balabonski, Courtieu, Pelle, Rieg, Tixeuil, Urbain)

Special locations:

- Base
- Companion

Invariant: connection



Assumptions

space, robots

- $\mathbb{R}^3 \rightsquigarrow \mathbb{R}^2$ flying over on a plane (!)
- Special point: base

Companion $\neq$ rescue team

- Volume/collision $\rightsquigarrow$ no mult. detection
- Vision bounded D_{max}
- Speed bounded D/round
- No Byzantine

Assumptions

execution

- FSYNC
- Movements rigid
- Initial config.: start from base

short cycles

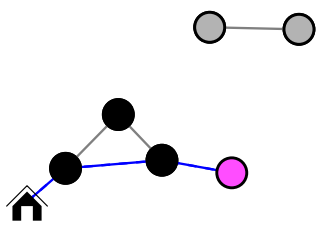
```
Parameter (n : nat).
Instance Robot_Names : Names := Robots n 0.

Parameter (D Dmax : R).
Instance Loc : Location := make_Location R2.
Instance SetObs := limited_set_observation Dmax.
Instance setting_is_rigid : RigidSetting.
```


Assumptions

execution, invariant 1

- FSYNC
- Movements rigid
- Initial config.: start from base

short cycles



Remain connected

Assumptions — execution, invariant 2

- FSYNC — short cycles
- Movements rigid
- Initial config.: start from base

No collision : cannot have two robots at same spot

Building a **correct** solution

1. Instance of model with assumptions [ok]
2. Strengthening of assumptions, towards sufficient set
3. Getting a (scheme of) correct protocol
4. (Providing a concrete protocol) (scheme $\neq \varnothing$)

Building a **correct** solution — strengthening

- Initial
- Orientation (prob. asymmetrical) — go towards companion **0**
- In flight — $\neq$ at base
- Contributing + ☠

```
Definition identifier := nat.
Definition launched := bool.
Definition alive := bool.
Definition state:= location * identifier * launched * alive
```


Building a **correct** solution — strengthening

Formal expression of invariants

```
Definition path_conf (cf:config) :=
∀ g, get_alive (cf g) = true →
 get_ident (cf g) = 0
 ∨ ∃ g', dist (get_loc (cf g)) (get_loc (cf g')) ≤ Dmax
                ∧ get_alive (cf g') = true
                ∧ get_launched (cf g') = true
                ∧ get_ident (cf g') < get_ident (cf g).
```

```
Definition no_collision_conf(cf:config) := ∀ g g', g ≠ g'
  → get_launched(cf g) = true → get_launched(cf g') = true
  → get_alive(cf g) = true → get_alive(cf g') = true
  → dist (get_loc (cf g)) (get_loc (cf g')) ≠ 0ℝ.
```

```
Definition NoCollAndPath e :=
forever (fun c ⇒ no_collision_conf c ∧ path_conf c) e.
```

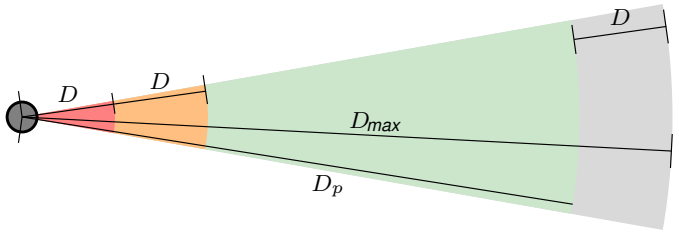
Building a **correct** solution — strengthening

Getting rid of trivial counter-examples

- Enough robots... (always a robot at base)
- Initial configuration valid: *companion connected and without collision*

Less trivial counter-examples

- Targetting a robot soon ☠? risk of losing connection... lights
- Cannot predict others' moves... defensive approach



⇝ Constraints : $D_{max} > 7D$ and launch at $D_{max} - 4D$ optimal?

Building a **correct** solution — candidate

1. Choose a target robot (direction)
2. Choose a destination
3. Safe?
   - Yes: go to destination, no warning
   - No: stay at location, display warning

Constraints over choices of 1) target and 2) destination

```
Axiom choose_new_pos_spec : ∀ obs target,
Definition protocole (s : observation) : R2*light :=
 let target := choose_target s in
 let new_pos := choose_new_pos s (fst target) in
 match move_to s new_pos with      (* Is this dangerous? *)
 | true ⇒ (new_pos,false)    (* Safe: move + light off. *)
 | false ⇒ ((0,0), true)   (* Danger: stay + light on.  *)
 end.
```


Building a **correct** solution — candidate

```
Axiom choose_target_spec : ∀ obs_id local_config,
   let obs := obs_from_config local_config in
   let target := choose_target obs_id obs in
      target ∈ obs              (* target must be in range *)
   ∧ get_alive target = true                  (* be alive *)
   ∧ get_ident target < get_ident obs_id (* smaller id *)
   ∧ (get_light target = true  (* preferably light off *)
        → ∀ id ∈ obs, get_light id = true)
   ∧ (get_light target = true       (* preferably close *)
        → dist (0,0) (get_loc target) > Dp
        → ∀ id ∈ obs, dist (0,0) (get_loc elt) > Dp).
```

```
Axiom choose_new_pos_spec : ∀ obs target,
  let new := choose_new_pos obs target in
    dist new target ≤ Dp    (* keep in range *)
  ∧ dist new (0,0) ≤ D.  (* reachable *)
```


Building a **correct** solution — suited family

Theorem `NoCollAndPath` invariant of any FSYNC execution, from a valid conf., of a candidate that fulfils these constraints.

- 520 lines of instantiation
- 540 lines of specifications
- 1000 lines of actual proof

No loss of contact with the rescue team!

Solution obtained

- within the formal framework
- along specification

Easy concrete solution (choices for target and dest. fulfilling constraints)

Future work

Towards a complete tool-chain for safe protocols

Background:

- New topologies, case studies, similar models (Coq)
- Comparison of demons

Core model:

- ASYNC: how to ease proofs?
- Randomized protocols

Tool chain:

- Automation
  - Links model-cheking/formal proof, Graphs and rewriting, WF
- Semantics $\rightsquigarrow$ DSL
  - Generation of proof obligations, phases, etc.

$\rightsquigarrow$ implementation. . .

A few words on. . .

SAPPORO = Research project on oblivious robots (France/Japan)

https://sapporo.liris.cnrs.fr/

Pactole = formal library for the Coq proof assistant
modelling robotic swarms https://pactole.liris.cnrs.fr/

Assets:

- single framework to express everything
- specification is easy (very close to math)
- proof of correctness + of impossibility
- compare expressive power of models

Examples:

- Space: ring, graph, plane, etc.
- Problems: gathering, convergence, exploration, lifeline