Formal certification of ASYNC protocols: Gathering at the Weber point$^{(s)}$

Maria-Virginia Aponte[3], Mathis Bouverot-Dupuis[1,3], **Quentin Bramas**[2], Pierre Courtieu[3], Lionel Rieg[4], and Xavier Urbain[5]

[1]ENS [2]University of Strasbourg [3]CNAM–Cedric [4]UGA–Verimag IUF [5]LIP6-UCBL1

slides available at https://bramas.fr

SIROCCO 2026

Context

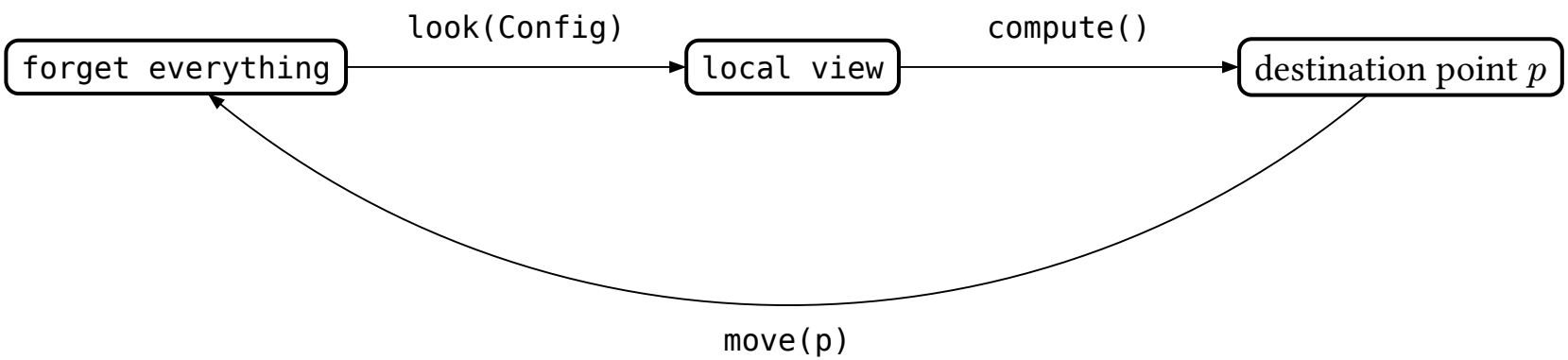
- Mobile robots in the continuous plane
- Formalizing ASYNC protocol with the Rocq/Pactole proof assistant
- A new algorithm solving the gathering problem in the ASYNC model

Mobile Robots

- points evolving in the $\mathbb{R}^2$ plane
- performing Look–Compute–Move cycles

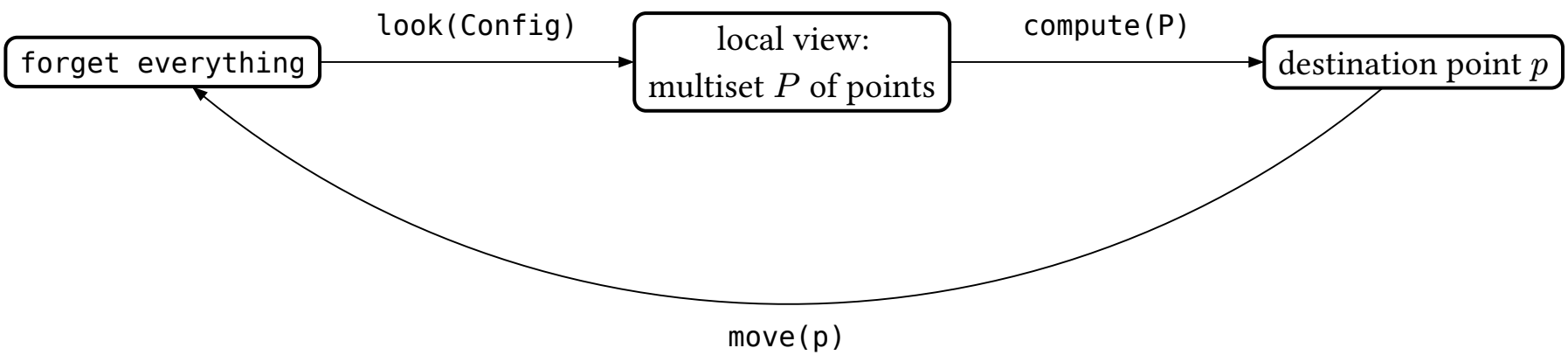
Mobile Robots

- points evolving in the $\mathbb{R}^2$ plane
- performing Look–Compute–Move cycles



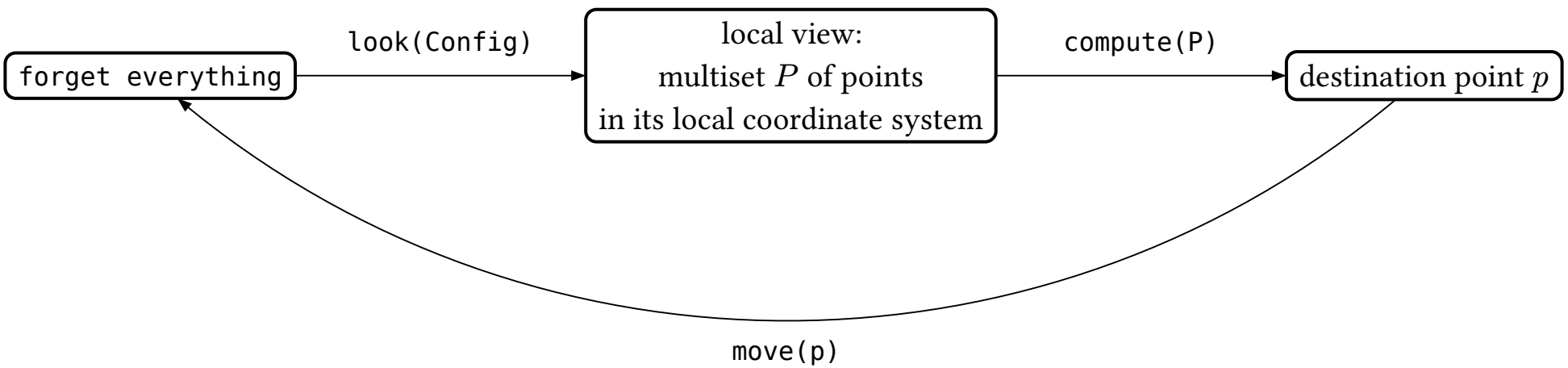
Mobile Robots

- points evolving in the $\mathbb{R}^2$ plane
- performing Look–Compute–Move cycles



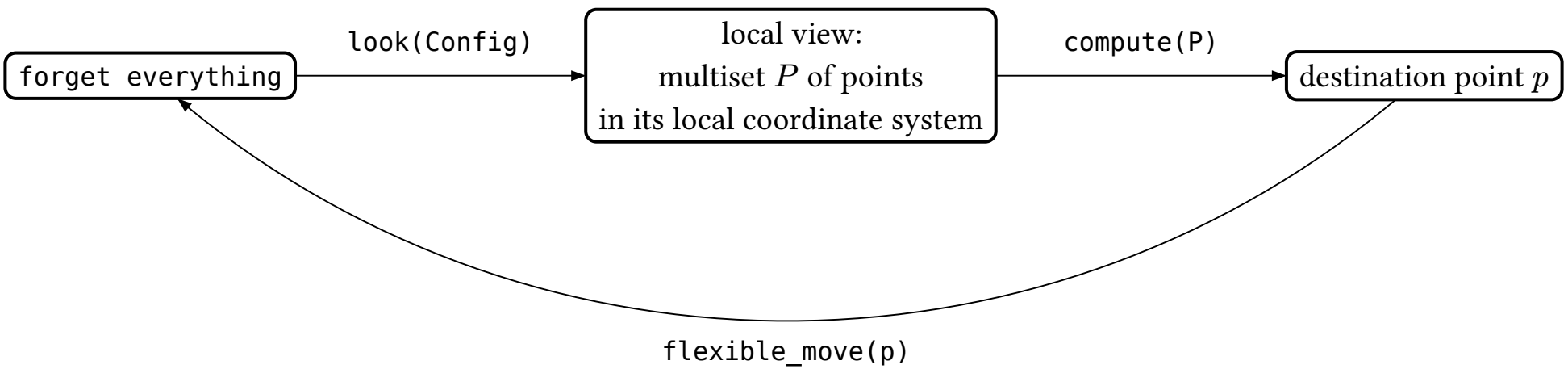
Mobile Robots

- points evolving in the $\mathbb{R}^2$ plane
- performing Look–Compute–Move cycles

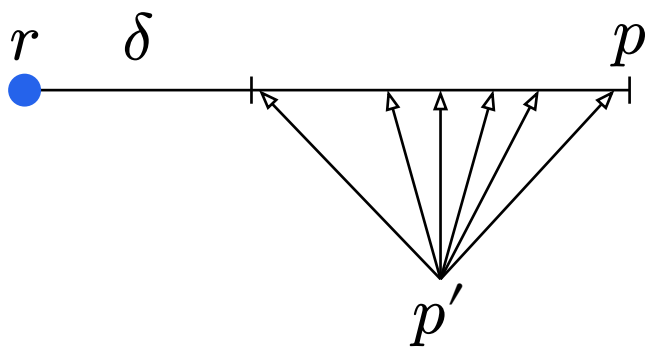


Mobile Robots

- points evolving in the $\mathbb{R}^2$ plane
- performing Look–Compute–Move cycles



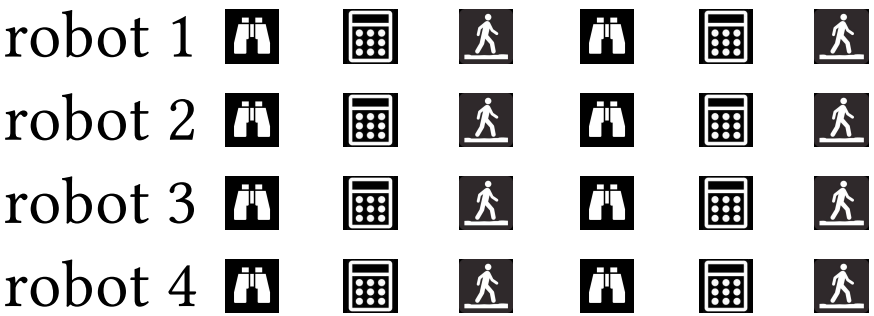
Flexible Moves



The demon can interrupt the move at any time, as long as the robot has moved at least δ toward p.

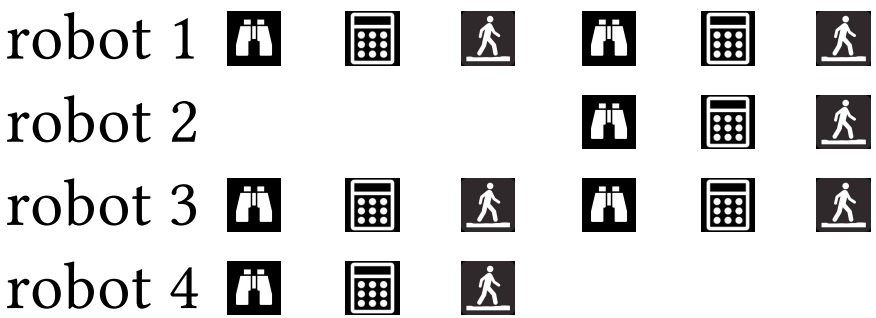
Synchrony

FSYNC



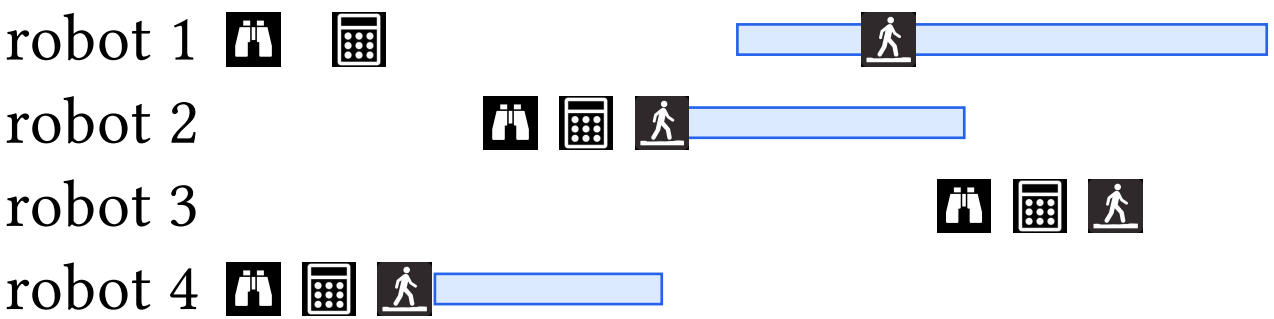
Synchrony

FSYNC , SSYNC



Synchrony

FSYNC , SSYNC , or ASYNC



Modeling an SSYNC Round in Pactole

Atomic Round:

- **select** the set A of activated robots
-

Modeling an SSYNC Round in Pactole

Atomic Round:

- **select** the set A of activated robots
- for each active robot r:
  - ▸

Modeling an SSYNC Round in Pactole

Atomic Round:

- **select** the set A of activated robots
- for each active robot r:
  - **select** an arbitrary coordinate system.
  -

Modeling an SSYNC Round in Pactole

Atomic Round:

- **select** the set A of activated robots
- for each active robot r:
  - **select** an arbitrary coordinate system.
  - build the multiset S of positions in this coordinate system.
  -

Modeling an SSYNC Round in Pactole

Atomic Round:

- **select** the set A of activated robots
- for each active robot r:
  - **select** an arbitrary coordinate system.
  - build the multiset S of positions in this coordinate system.
  - call the algorithm with S, obtain destination d.
  -

Modeling an SSYNC Round in Pactole

Atomic Round:

- **select** the set A of activated robots
- for each active robot r:
  - **select** an arbitrary coordinate system.
  - build the multiset S of positions in this coordinate system.
  - call the algorithm with S, obtain destination d.
  - **select** a point between the current position and the destination
  -

Modeling an SSYNC Round in Pactole

Atomic Round:

- **select** the set A of activated robots
- for each active robot r:
  - **select** an arbitrary coordinate system.
  - build the multiset S of positions in this coordinate system.
  - call the algorithm with S, obtain destination d.
  - **select** a point between the current position and the destination
  - update the position
-

Modeling an SSYNC Round in Pactole

Atomic Round:

- **select** the set A of activated robots
- for each active robot r:
  - **select** an arbitrary coordinate system.
  - build the multiset S of positions in this coordinate system.
  - call the algorithm with S, obtain destination d.
  - **select** a point between the current position and the destination
  - update the position
- **do not move inactive robot**

Modeling an ASYNC Round in Pactole

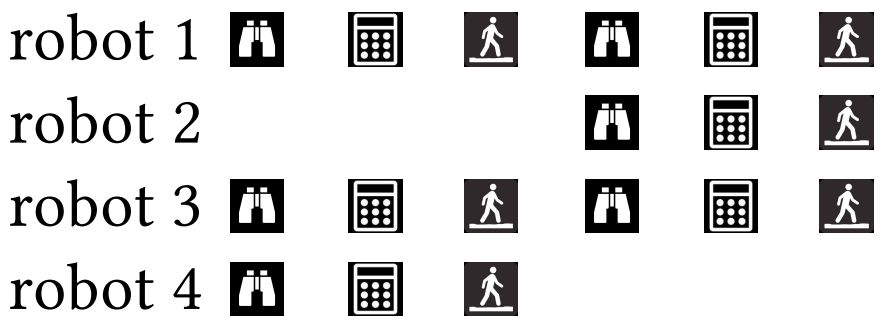
Goal: a Pactole instantiation that faithfully represents the Suzuki–Yamashita execution model in its flexible ASYNC variant.

- continuous time, but robots are activated only at specific instants
- between two instants: not observed → not modeled
- the simulator jumps from one instant to the next, that is, from one round to the next
- the formalization must be validated for soundness and completeness

Modeling an ASYNC Round in Pactole

Observing a robot position:

- SSYNC: necessarily at its last computed destination



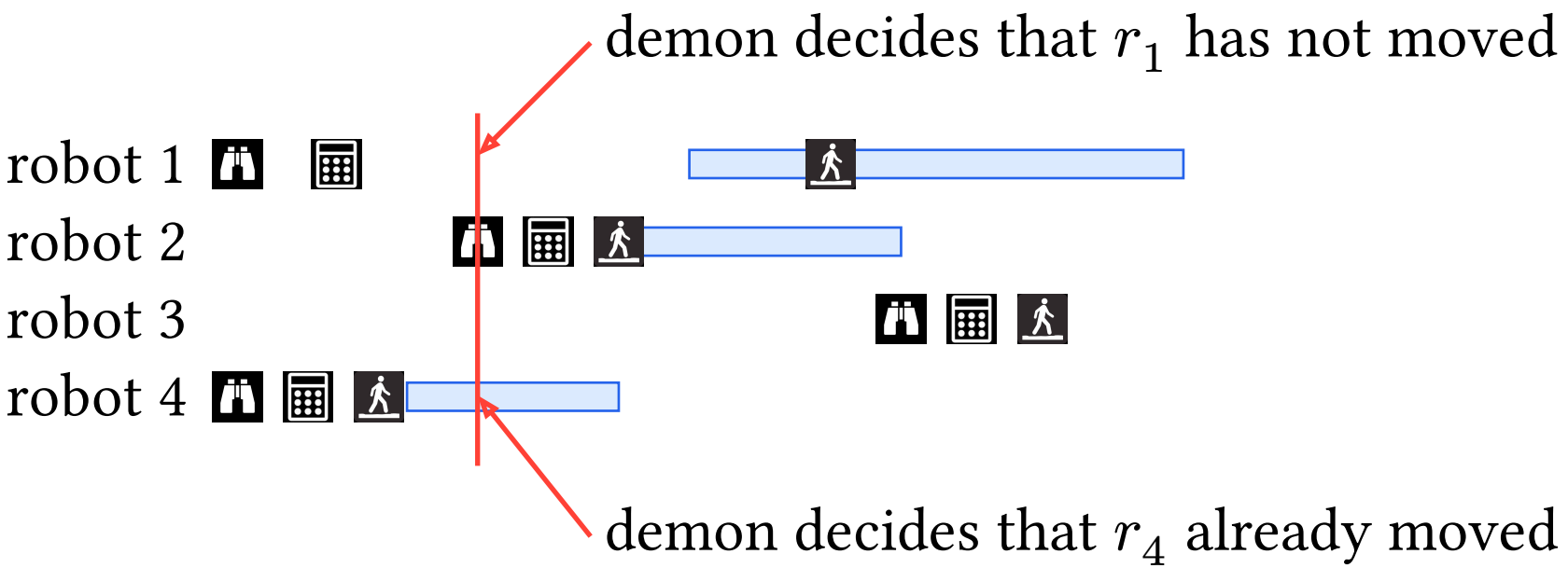
Modeling an ASYNC Round in Pactole

Observing a robot position:

- SSYNC: necessarily at its last computed destination
- ASYNC: somewhere on the path between its last observed position and its destination

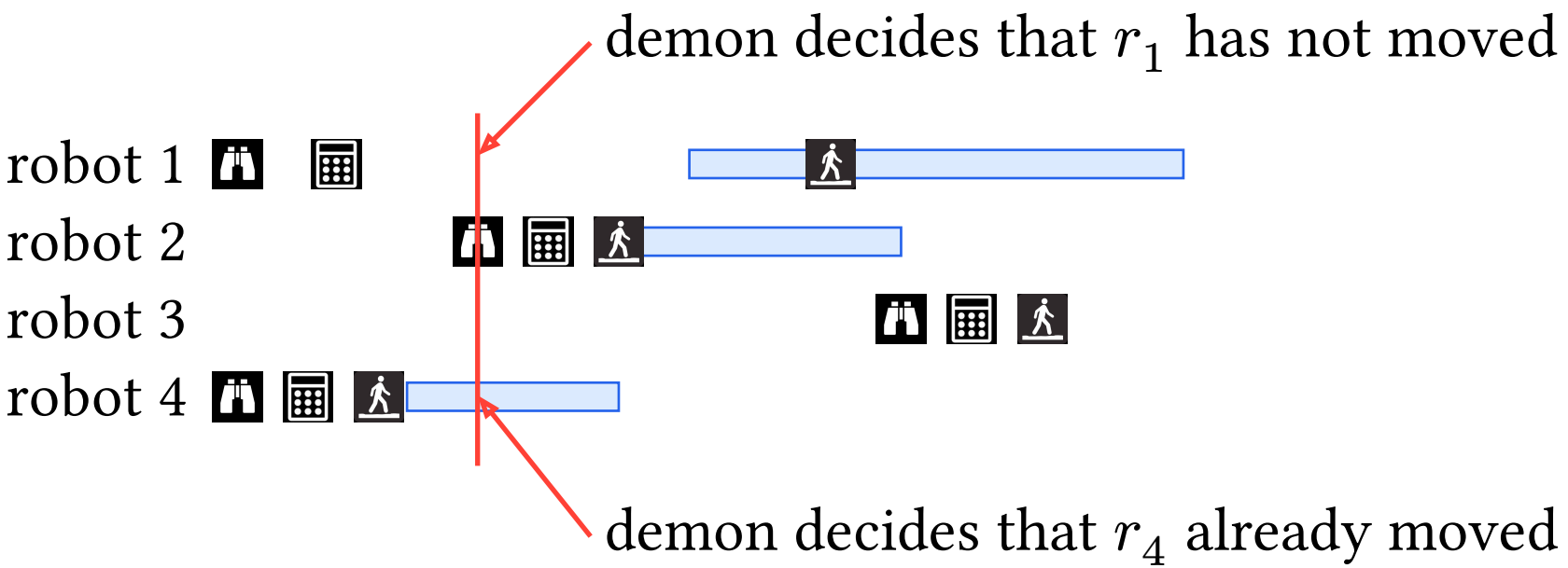
Modeling an ASYNC Round in Pactole

Observing a robot position:



Modeling an ASYNC Round in Pactole

Observing a robot position:



After this round, the robots can/must continue moving even if inactive.

Modeling an ASYNC Round in Pactole

- therefore the configuration must store the missing information
- but the observation still exposes only what a robot is allowed to perceive

Chosen Formalization

Robot state

$\{\text{start} : \mathbb{R}^2; \text{target} : \mathbb{R}^2; \text{ratio} : \mathbb{R}\}$

$\text{location}(\text{st}) = \text{st.start} + \text{st.ratio} * (\text{st.target} - \text{st.start})$

Inactive robot

$$\text{upd\_loc}\ (\text{st}, \rho) = \text{st.start} + \min(1, \text{st.ratio} + \rho) * (\text{st.target} - \text{st.start})$$

Active robot

- compute a new target
- reset the ratio to 0

Chosen Formalization

rocq

```
Variables δ: R.
Hypothesis δ_g0: (0 < δ).
Definition state := location * location * ratio.
Definition get_location := fun (st, dst, r) ⇒ (st + r * (dst-st)).

Definition flex_da_prop(da:demonic_action) := ∀ id cfg, activated da id
    → get_location (cfg id) ≡ get_dest (cfg id)
      ∨ (δ ≤ dist (get_start (cfg id)) (get_location (cfg id))).

Definition update c g _ target _ := (get_location (c g), target, 0).
Definition inactive c id ρ := let '(s,d,r) := c id in (s,d,min(1,r+ρ)).
```

The Weber Point

Definition (Sum of distances to a point): $D_X(p) = \sum_{x \in X} d(p, x)$

Definition (Weber point):

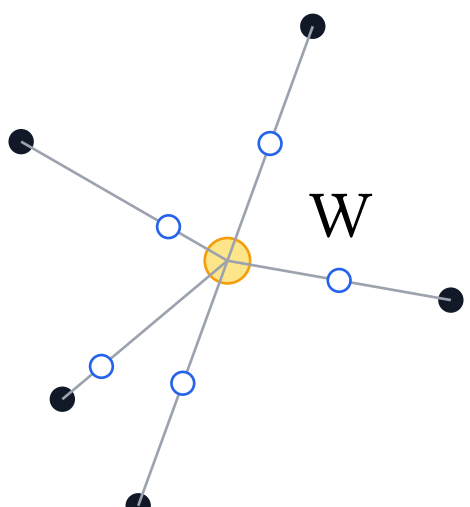
p is a Weber point of X iff it minimizes D_X

rocq

```
1 Definition dist_sum pts (x:R2) := list_sum (List.map (dist x) pts).
2 Definition argmin {A: Type} (f: A -> R): A -> Prop :=
3   fun a => forall b, (f a <= f b)%R.
4 Definition Weber pts: R2 -> Prop := argmin (dist_sum pts).
5 Definition OnlyWeber pts w: Prop :=
6   Weber pts w /\ (forall x, Weber pts x -> x == w).
```

Contraction Property

If a multiset of points contracts toward a Weber point W, then W remains a Weber point after the contraction.
In the unique case, uniqueness is preserved as well.



Contraction Property

```rocq
Lemma weber_contract ps ps' w:
  contract ps ps' w -> Weber ps w -> Weber ps' w.

Lemma weber_contract_unique ps ps' w:
  contract ps ps' w -> OnlyWeber ps w -> OnlyWeber ps' w.
```

Uniqueness of the Weber Point

Theorem: If the points are not all aligned, then the Weber point is unique.

```
Lemma weber_Naligned_unique points w:                                    rocq
  ~aligned points -> Weber points w -> OnlyWeber points w.
```

Characterization of the Aligned Case

Theorem: For every multiset X of aligned points:

- p is a Weber point iff

$$|\#\text{left}(X) - \#\text{right}(X)| \leq \#\text{on}(X)$$

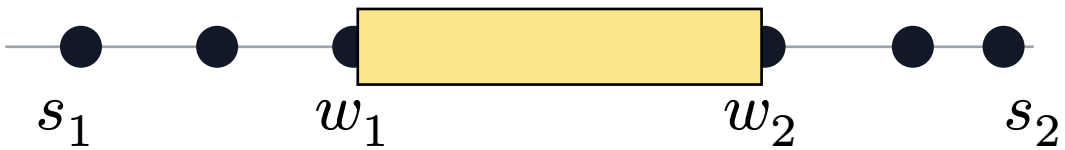
- p is the unique Weber point iff

$$|\#\text{left}(X) - \#\text{right}(X)| < \#\text{on}(X)$$

Unique Aligned Example

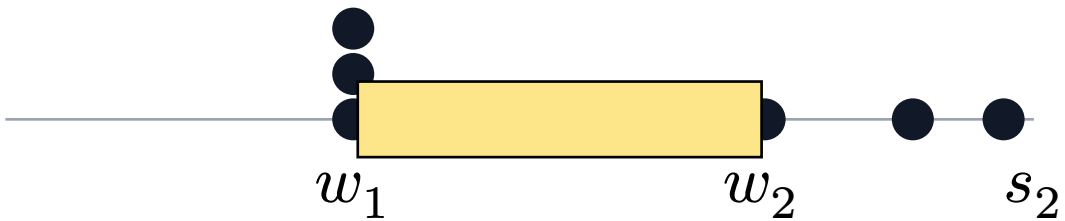
- multiplicity at $w = 1$
- left-right difference $= 0$
- hence w is the unique Weber point

Non-Unique Example



- w_1 is a Weber point
- w_2 is a Weber point
- every point in $[w_1, w_2]$ is a Weber point

Non-Unique Example with Multiplicity



Gathering

- The usual accepted solution,

move cautiously toward *the* Weber point

is ill-defined.

Gathering

- The usual accepted solution,

move cautiously toward *the* Weber point

is ill-defined.

- And the variant

move cautiously toward *a* Weber point

does not solve gathering.

Gathering

- The usual accepted solution,

move cautiously toward *the* Weber point

is ill-defined.

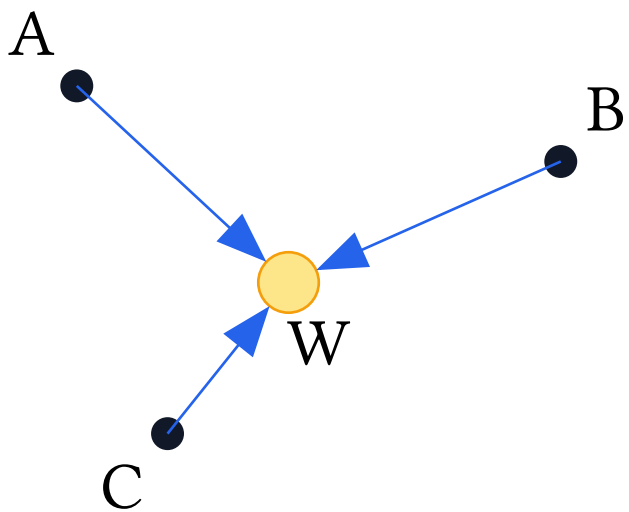
- And the variant

move cautiously toward *a* Weber point

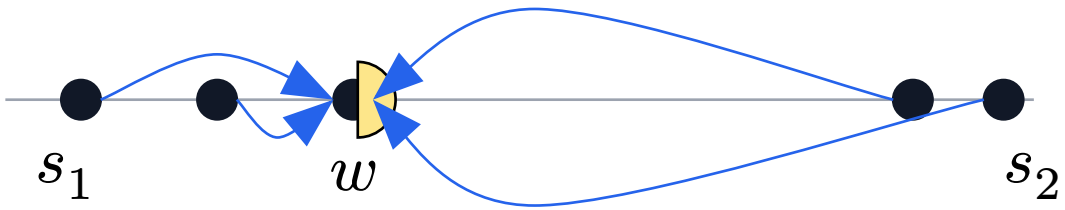
does not solve gathering.

→ Therefore the non-unique case requires a dedicated treatment.

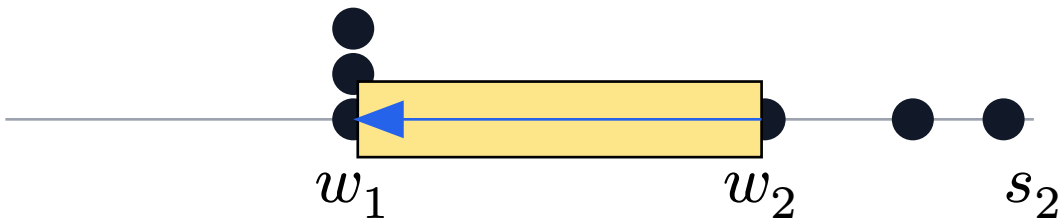
Algorithm: Non-Aligned Case



Algorithm: Aligned Case & Unique Weber Point

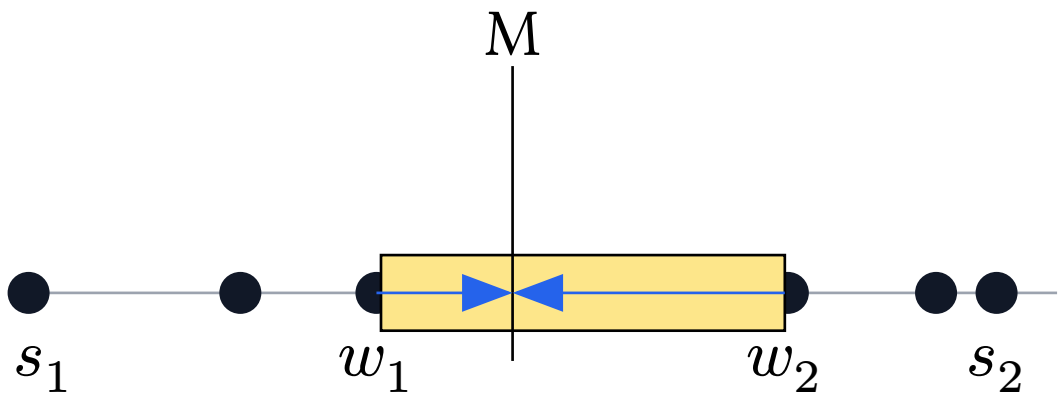


Aligned Case & Majority Tower

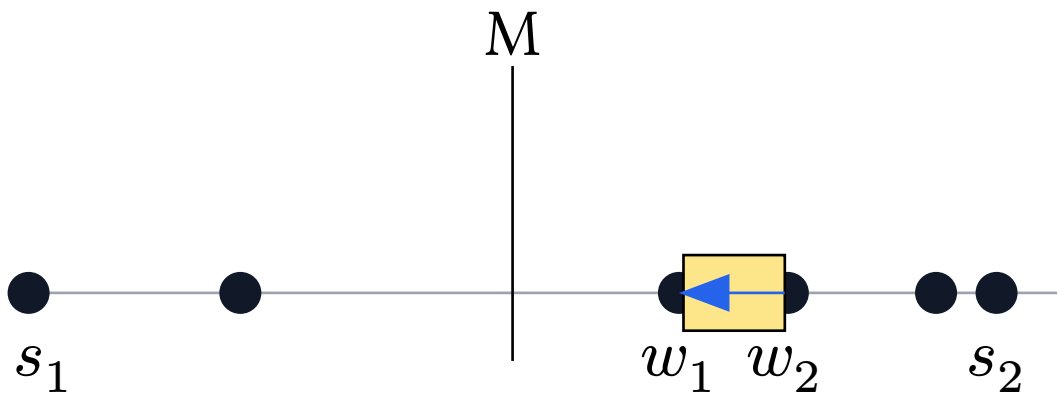


Cautious move toward the majority tower.

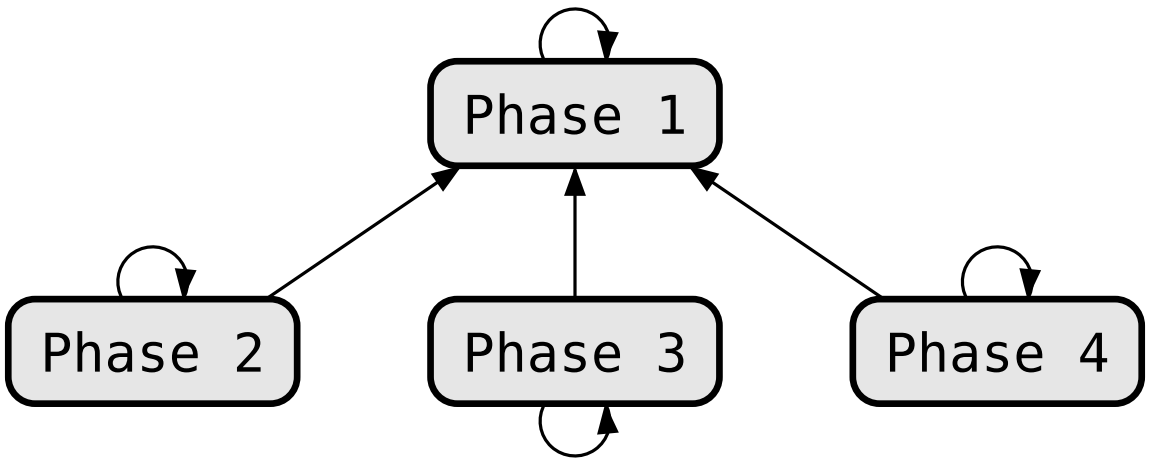
Aligned & Non-Unique WP & No Majority Tower (1)



Aligned & Non-Unique WP & No Majority Tower (2)

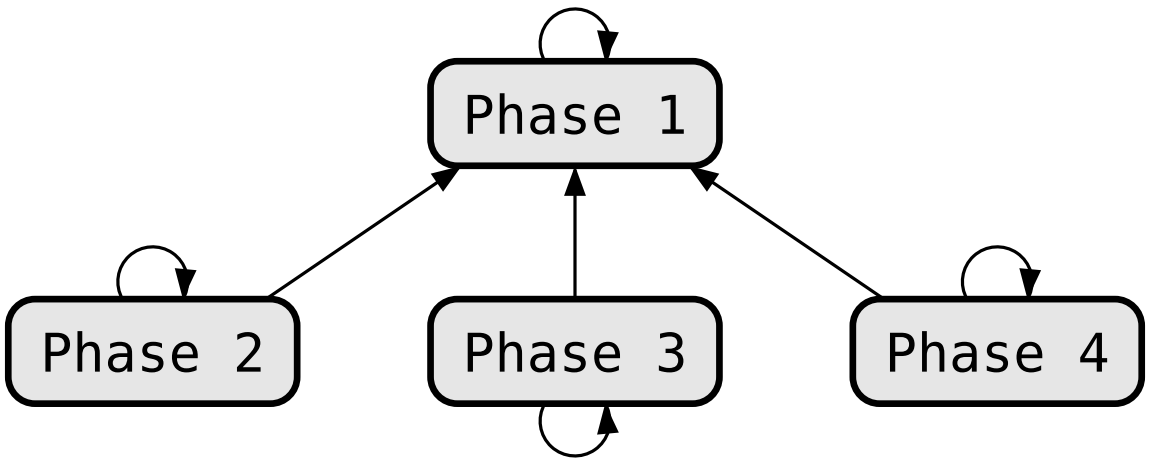


Phases



- 1: unique Weber point
- 2: $\neg 1 \wedge$ majority_tower
- 3: $\neg 1 \wedge \neg 2 \wedge M \in [w_1, w_2]$
- 4: $\neg 1 \wedge \neg 2 \wedge \neg 3$

Phases



- 1: unique Weber point
- 2: $\neg 1 \wedge$ majority_tower
- 3: $\neg 1 \wedge \neg 2 \wedge M \in [w_1, w_2]$
- 4: $\neg 1 \wedge \neg 2 \wedge \neg 3$

- the graph is exhaustive
- no cycle can be traversed infinitely often

Final Theorem

```rocq
Theorem gather_correct c d:
  Fair d ->
  (Stream.forever (Stream.instant similarity_da_prop)) d ->
  (Stream.forever (Stream.instant flex_da_prop)) d ->
  ~invalid c ->
  config_stay c ->
  WillGather (execute gatherW d c).
```

Under the exact model assumptions, executions eventually gather and remain gathered forever.

Conclusion

Formalization of the ASYNC model in Pactole, and proof of a non-trivial algorithm

Conclusion

Formalization of the ASYNC model in Pactole, and proof of a non-trivial algorithm

- topology $\mathbb{R}^2$, straight-line moves
- fair, flexible ASYNC demon
- robots:
  - are oblivious, disoriented, and equipped with multiplicity detection
  - compute exactly over the reals
  - can compute the Weber segment of a multiset of points
  - are initially idle
- the initial configuration is non-bivalent

All the proofs + Pactole lib: archive linked in the paper.

Conclusion

Formalization of the ASYNC model in Pactole, and proof of a non-trivial algorithm

- topology $\mathbb{R}^2$, straight-line moves
- fair, flexible ASYNC demon
- robots:
  - are oblivious, disoriented, and equipped with multiplicity detection
  - compute exactly over the reals
  - can compute the Weber segment of a multiset of points
  - are initially idle **Open problem: what if they are initially moving?**
- the initial configuration is non-bivalent

All the proofs + Pactole lib: archive linked in the paper.

Conclusion

Formalization of the ASYNC model in Pactole, and proof of a non-trivial algorithm

- topology $\mathbb{R}^2$, straight-line moves
- fair, flexible ASYNC demon
- robots:
  - are oblivious, disoriented, and equipped with multiplicity detection
  - compute exactly over the reals
  - can compute the Weber segment of a multiset of points
  - are initially idle **Open problem: what if they are initially moving?**
- the initial configuration is non-bivalent

All the proofs + Pactole lib: archive linked in the paper.

Thank you

– end –

Can llm be used to do it ?

Of course they can help, with the syntax and when you are inside an "easy" proof.

But they will struggle with "impossible" proofs (ie, when you write wrong statements).

Can my student do it ?

Of course they can help, with the syntax and when you are inside an "easy" proof.

But they will struggle with "impossible" proofs (ie, when you write wrong statements).

how many lines of code ?

- Weber points
  - 1150 lines of specifications
  - 4500 lines of proofs.
- Regarding the case study, that is including the specifications, intermediate lemmas, and the proof for the main theorem
  - 500 lines of specifications
  - 2900 lines of proofs only.