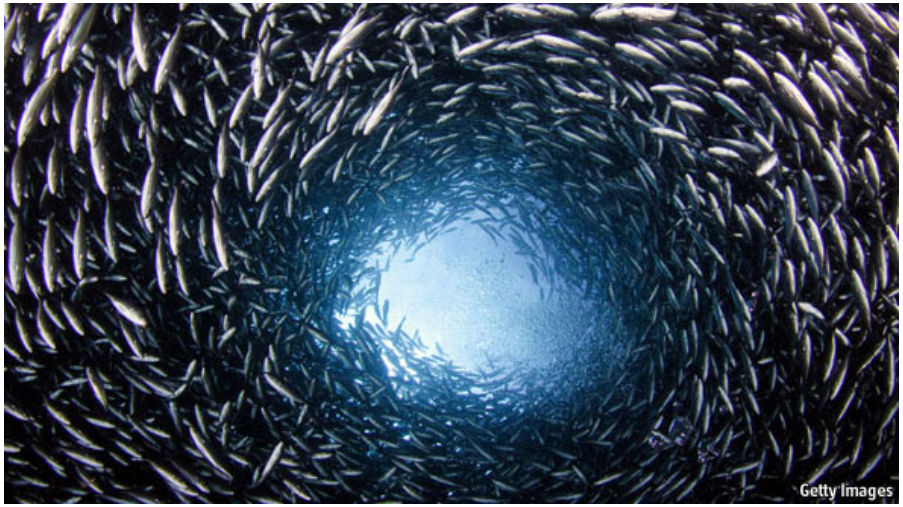
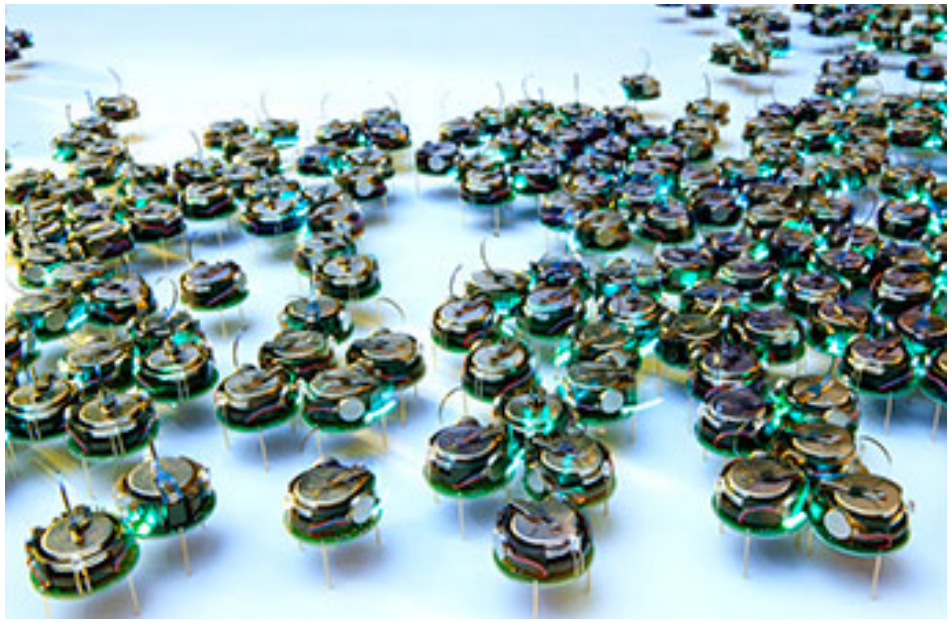
Multi-Agent (Smart) Systems with Virtual Stigmergies

Rocco De Nicola[1] **Luca Di Stefano**[2] Omar Inverso[2]

[1] IMT School of Advanced Studies, Lucca, Italy
[2] Gran Sasso Science Institute (GSSI), L'Aquila, Italy

Systems of agents





Sources of complexity

- Large number of agents
- Open-endedness
- Asynchronous interaction

Emerging Behaviour

Vision

- High-level **language**
- Structure-preserving **encoding**
- Symbolic **verification**
- Structure-aware **decision procedures**

- **Concepts** from the MAS community
- **Methodologies** from the FM/verification communities

Vision

- High-level **language**
- Structure-preserving **encoding**
- Symbolic **verification**
- Structure-aware **decision procedures**

- **Concepts** from the MAS community
- **Methodologies** from the FM/verification communities

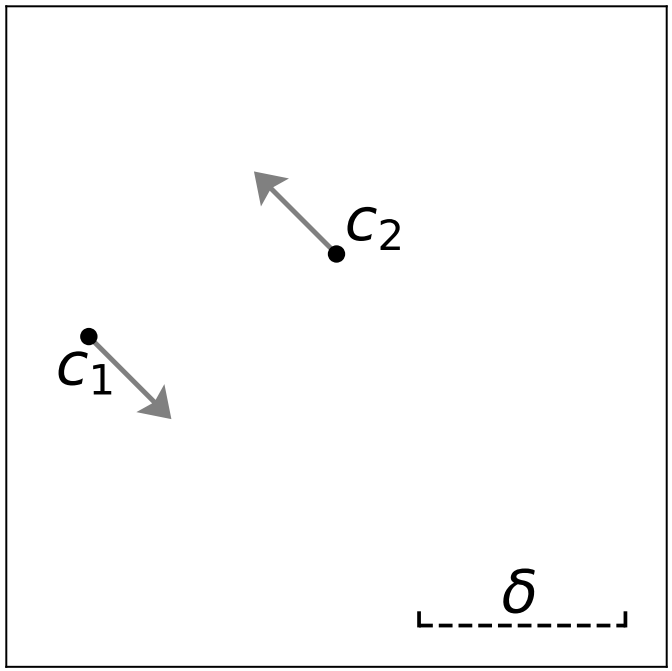
Language: LAbS

- A Language with Attribute-based Stigmergies
- Stimergies: indirect propagation of knowledge
- Inspired by biologic systems
- Attribute-based: language user can configure the interaction mechanism

Virtual Stigmergy: Example (1/2)

- Two agents c_1, c_2 in a $G \times G$ arena
- Each agents moves in a *direction* (d_x, d_y), stored in the stigmergy
- Attribute-based predicate: interaction may happen if distance $< \delta$

$$P \triangleq x, y \leftarrow x + d_x \bmod G, y + d_y \bmod G; P$$

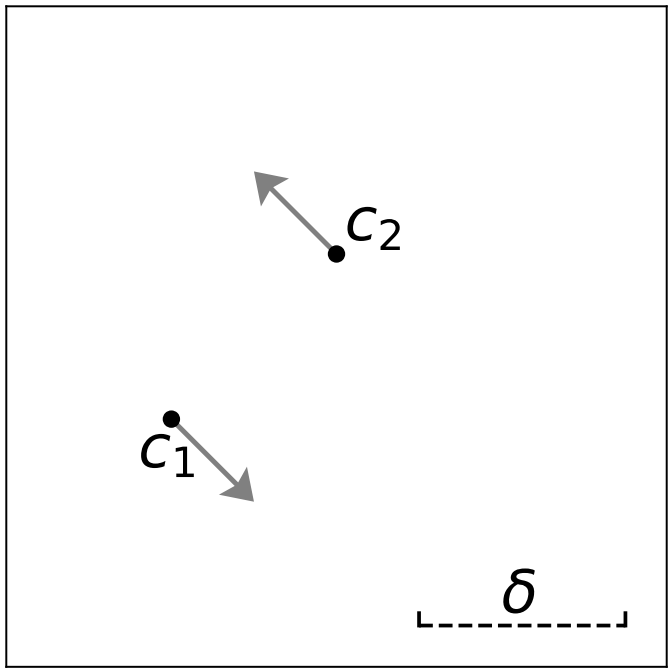


Initial state

Virtual Stigmergy: Example (1/2)

- Two agents c_1, c_2 in a $G \times G$ arena
- Each agents moves in a *direction* (d_x, d_y), stored in the stigmergy
- Attribute-based predicate: interaction may happen if distance $< \delta$

$$P \triangleq x, y \leftarrow x + d_x \bmod G, y + d_y \bmod G; P$$

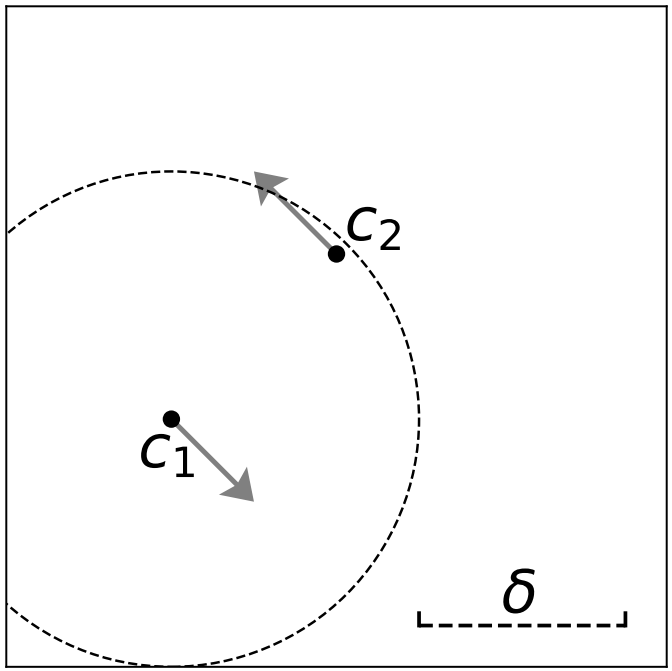


c_1 reads (d_x, d_y) and moves

Virtual Stigmergy: Example (1/2)

- Two agents c_1, c_2 in a $G \times G$ arena
- Each agents moves in a *direction* (d_x, d_y), stored in the stigmergy
- Attribute-based predicate: interaction may happen if distance $< \delta$

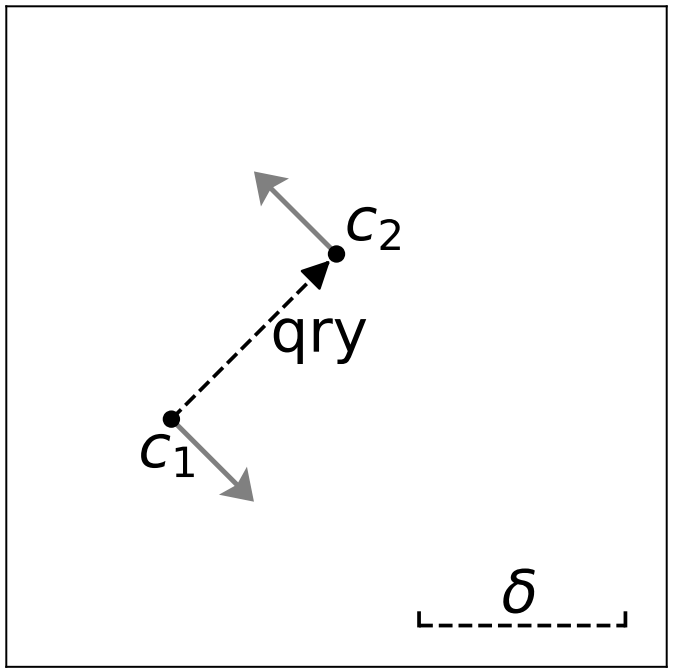
$$P \triangleq x, y \leftarrow x + d_x \bmod G, y + d_y \bmod G; P$$



Virtual Stigmergy: Example (1/2)

- Two agents c_1, c_2 in a $G \times G$ arena
- Each agents moves in a *direction* (d_x, d_y), stored in the stigmergy
- Attribute-based predicate: interaction may happen if distance $< \delta$

$$P \triangleq x, y \leftarrow x + d_x \bmod G, y + d_y \bmod G; P$$

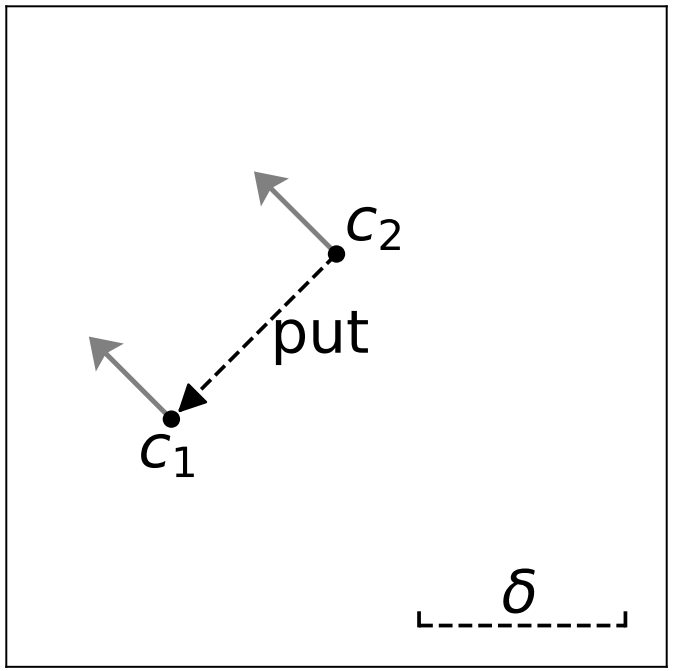


qry: agent asks for confirmation on (d_x, d_y)

Virtual Stigmergy: Example (1/2)

- Two agents c_1, c_2 in a $G \times G$ arena
- Each agents moves in a *direction* (d_x, d_y), stored in the stigmergy
- Attribute-based predicate: interaction may happen if distance $< \delta$

$$P \triangleq x, y \leftarrow x + d_x \bmod G, y + d_y \bmod G; P$$



put: c_2 propagates his (newer) value

Encoding

Multi-robot systems
Gossiping protocols
Population protocols
Flocking, foraging, ...
...

Encoding

Under-approximation
- Bounded MC
- Symbolic execution
- Statistical MC, ...

Over-Approximation
- Abstract interpretation
- Predicate abstraction, ...

Unbounded analysis
- K-induction
- IC3, ...

- Compact
- Embedded scheduler

Encoding

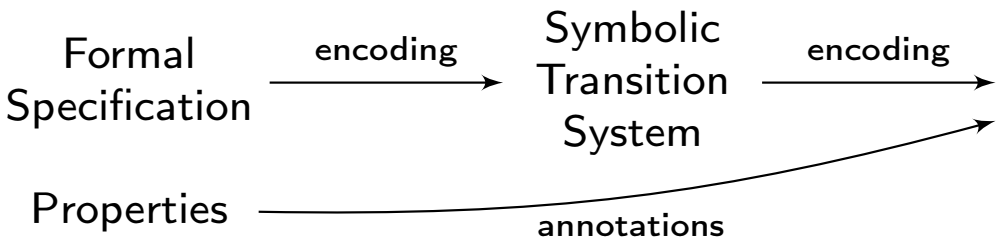
Multi-robot systems
Gossiping protocols
Population protocols
Flocking, foraging, ...
...

Encoding

Under-approximation
- Bounded MC
- Symbolic execution
- Statistical MC, ...

Over-Approximation
- Abstract interpretation
- Predicate abstraction, ...

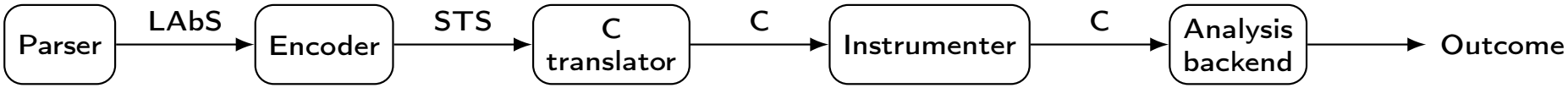
Unbounded analysis
- K-induction
- IC3, ...



- Compact
- Embedded scheduler

SLiVER

Symbolic LAbS Verification[1]



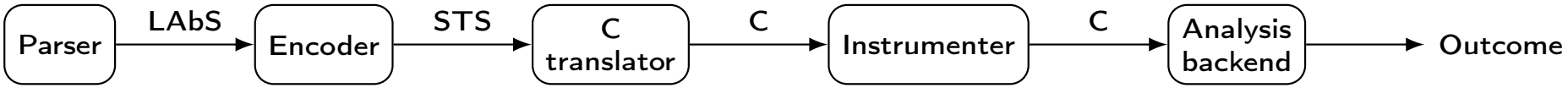
Aim: Push-button analysis
Modularity

- Backend
- Target language
  - C
  - LNT: explicit MC, simulation[2]

[1] https://github.com/labs-lang/sliver/
[2] CADP, https://cadp.inria.fr/

SLiVER

Symbolic LAbS Verification[1]



Aim: Push-button analysis
Modularity

- Backend
- Target language
  - C
  - LNT: explicit MC, simulation[2]

[1] https://github.com/labs-lang/sliver/
[2] CADP, https://cadp.inria.fr/

Preliminary results: unbounded analysis

- Safety: Absence of assertion violations
- Liveness: Eventual reachability (encoded as termination)

System	Property	Bounded model checking[3]	Summarization	IC3	Symbolic execution	Predicate analysis
formation-safe	L	•	?	–		–
formation-safe	S	–	•	•	•	•
flock-safe	L		•	–	?	–
majority-safe	S	–	?	•		•
majority-safe	L		?	–	•	–

- • Correct result
- – Not supported
- ? Analysis inconclusive
- (empty) Out of time/memory

[3](with completeness threshold)

Future work

- Correctness of the encoding
- Structure-aware reasoning
- Unbounded number of agents (inductive proofs?)
- Completeness threshold (for under-approximation)
- Distribute/speed up analysis

References

1. R. De Nicola, L. Di Stefano, and O. Inverso, *"Toward Formal Models and Languages for Verifiable Multi-Robot Systems,"* Front. Robot. AI, vol. 5, 2018.
2. R. De Nicola, L. Di Stefano, and O. Inverso, *"Multi-agent Systems with Virtual Stigmergy,"* to appear in Sci. Comput. Program.
3. R. De Nicola, L. Di Stefano, and O. Inverso, *"Verification of Multi-agent Systems with Stigmergic Interaction,"* under submission.

C encoding sketch (1/2)

```
N = ...   // Number of agents
K_I = ... // Number of attributes
K_L = ... // Number of stigmergy keys
B = ...   // Number of transitions

int v[N];                                   // v[i] --> program counter for agent i
int I[N][K_I], Lvalue[N][K_L], Ltstamp[N][K_L];    // integer values for all keys
bool Zc[N][K_L], Zp[N][K_L]    // Zc[i][j]=1 --> key j is in Zc of agent i (or Zp)

/* System-level actions */
void attr(int id, int key, int value) { ... }   // encodes "key <- value"
void lstig(int id, int key, int value) { ... }  // encodes "key <~ value"
bool link(int a, int b, int key) { ... } // link predicate true iff.  I_a, L_a, I_b, L_b ⊨ φ_key
void confirm(void) { ... }
void propagate(void) { ... }

/* Agent-level actions */
void stmt0(int tid) { ... }
void stmt1(int tid) { ... }
// ...

/* Initialisation and properties to verify */
void init() { ... }
void monitor(void) { assert ( φ ) }
void check(void) { if ( φ ) exit(); }
```

C encoding sketch (2/2)

```
int nextAgent(int agent) { ... } // Scheduler assumptions

/* Scheduler */
int main(void) {
  init();
  int agent = *;
  assume(agent < N);

  while (1) {   // execution loop
    if ( * ) {
      switch (choice) {
        case 0: stmt0(agent);
        case 1: stmt1(agent);
        // ...
      }
    }
    else {
      if ( * ) propagate();
      else confirm();
    }

    monitor();
    check();
    agent = nextAgent(agent);
  }


}
```